MICRO GUIAS

MSX

de la A a la Z



microguías MSX

Explicación detallada de todas las palabras relacionadas con el sistema MSX



CARRETERA DE CANILLAS, 144 - 28043 MADRID
TELEFS. (91) 200 97 46/47

Título de la obra:

MICROGUIAS MSX

Depósito Legal: M. 6583 - 1986
I. S. B. N.: 84 - 86381 - 10 - X
Imprime: Signo Impresores, S. A.
Albasanz, 27. 28037 MADRID

Publicado por RA-MA

Carretera de Canillas, 144
28043 Madrid
España.

Equipo de Redacción

Dirigido por:
Angel L. Andeyro

Colaboradores:
Miguel y Francisco Perez de la Blanca
Pedro Gato

PROLOGO

Con la aparición del sistema MSX se produjo un hito histórico en el desarrollo de los microordenadores domésticos. Por primera vez, una serie de compañías de renombrado prestigio se ponían de acuerdo para fabricar sus aparatos bajo unas condiciones estandar.
Estas condiciones determinan, de hecho, la posibilidad de intercambio de programas y periféricos, es decir, la compatibilidad a todos los niveles, entre los distintos fabricantes.
Las especificaciones se basan en un microprocesador muy conocido, el Z-80 en su versión A, que permite una mayor velocidad de funcionamiento, un procesador de Video (VDP) que dispone de memoria de video independiente, un generador de sonido (PSG) AY-3-8910, con posibilidad de generación de tres canales de sonido diferentes, en tres octavas y un interfaz para periféricos (PPI) 8255 para el control del teclado, cassette, etc.
Junto a estos circuitos se encuentra una memoria RAM mínima de 8K, pero que en la práctica alcanza los 64K.
El lenguaje utilizado es el MSX-BASIC, una versión de Microsoft, que ocupa junto con el Sistema Operativo un total de 32K en ROM, más un área de trabajo de 3K, lo que deja unos 29K libres para usuario, del total de 64K direccionables simultaneamente por el microprocesador.
Esta versión es especialmente potente, al disponer además de las instrucciones usuales, de otras para tratamiento de archivos, control de sprites, trazado de gráficos mediante un microlenguaje especial, composición musical, etc., siendo semejante

al de otros ordenadores de mayor tamaño.
El sistema también admite la utilización de discos flexibles, para lo que dispone de un sistema operativo de disco propio el MSX-DOS, en disco o en cartucho, que complementa el sistema operativo residente en ROM con todas las operaciones de disco. Asímismo, existe una versión de BASIC para disco.
El estandar admite también otros sistemas operativos, como el CP/M en su versión 2.2.
La colección de Microguías, en este volumen dedicado al sistema MSX, recoje todas aquellas expresiones relacionadas con el mismo, especialmente las instrucciones, funciones y operaciones del lenguaje BASIC y del lenguaje máquina Z-80.
También se han incluido las órdenes del sistema operativo MSX-DOS y otros términos de uso general en informática.
Como primicia se esbozan las características de la próxima generación de equipos MSX2, en fase prototipo.
Se ha seguido el orden alfabético para permitir un fácil acceso, habiéndose marcado las iniciales de las palabras en cada página.
Cuando los términos son palabras clave reconocidas por el ordenador (sea en BASIC o en lenguaje ensamblador) se encuentran impresas en mayúsculas.
Cuando se trata de términos de uso general que no guardan relación con el sistema, se han escrito en minúsculas (con la excepción de las abreviaturas como BASIC, ASCII, etc.).
Dentro del texto o al final de la explicación suelen figurar referencias de las palabras asociadas, cuya lectura se aconseja.
Siempre que ha sido posible, se han incluido instrucciones en modo directo o pequeñas rutinas

para mostrar la acción de cada instrucción, función y operación, cuya ejecución es recomendable, junto con todas las posibles opciones y variantes que el lector pueda imaginar.
Con esta Microguía se intenta poner en manos del usuario y programador en MSX, de un medio de consulta rápida que le permita resolver sus dudas sobre la programación del equipo y sobre conceptos referentes al mismo que tenga que manejar.

Madrid, Febrero de 1.986

ABS Una función BASIC que calcula el valor absoluto de una expresión numérica, eliminando el signo negativo. El argumento debe ir entre paréntesis.

```
PRINT ABS(3)
PRINT ABS(-3)        dan como resultado 3.
```

acumulador El principal registro del procesador Z-80. Se representa por la letra A, y se usa normalmente con la mayoría de las operaciones, para transferir, sumar o ejecutar instrucciones.Su nombre procede de la forma en que opera, almacenando el resultado en sí mismo. Este registro puede almacenar números entre 0 y 255 al estar formado por 8 bits. (Véase registro).

ADC Un mnemónico de instrucción Z-80, significa suma con exceso. Opera con el acumulador o el par de registros HL. El exceso se representa mediante un bit del registro de bandera. La bandera adquiere el nivel 1 si la suma es demasiado grande para almacenarse en el acumulador. Al ejecutarse, el operando más el exceso se suman al contenido del acumulador.

```
ADC A,4        suma 5 a A si la bandera de exce-
               so es 1.
```

ADD Un mnemónico de instrucción Z-80. Significa suma. Opera con el acumulador o el par de registros HL (8 y 16 bits, respectivamente). Para sumar dos valores, el registro A ó HL se cargan con el primer valor y después se les suma el segundo.

```
LD A,5
ADD A,6
```

Es muy flexible cuando se utiliza el registro A,

pues puede operar con datos, contenidos de registros, contenido de posiciones de memoria y registros de índice.

aleatorio Significa al azar. El ordenador dispone de una función para la generación de números aleatorios, denominada RND.

algoritmo El procedimiento seguido para la resolución de un problema determinado, está constituido por una serie de pasos, que realizados ordenadamente, conducen a la solución del problema Es una fase intermedia entre el problema y la creación de un programa que lo resuelva.

alimentacion La alimentación del ordenador se realiza a partir de la red eléctrica. Dicha alimentación debe ser convenientemente transformada, rectificada y filtrada. Este proceso produce una cierta cantidad de calor que es disipado por el ordenador. Un corte de electricidad supone la pérdida de toda la información contenida en el ordenador, almacenada de forma no permanente.

almacenamiento Proceso por el cual la información se guarda para su uso posterior. Los sistemas de almacenamiento principales son: la memoria, la cinta de cassette, los discos magnéticos y los cartuchos.

AND (1) Un operador lógico binario de BASIC, que utiliza como operandos expresiones lógicas o numéricas. En el primer caso, se emplea la siguiente tabla de verdad:

X	Y	X AND Y
V	V	V
V	F	F
F	V	F
F	F	F

En el segundo caso, los operandos deben estar comprendidos entre 0 y 65536, descomponiéndose en sus bits constitutivos y operándose bit a bit según la anterior tabla de verdad (sustituyéndose el valor verdadero por 1, y el falso por 0), reconstruyéndose el resultado en notación decimal.

(X < 3) AND (Y > 5) será verdadero solamente si X es menor que 3 e Y es mayor que 5.

250 AND 28 = 11111010 AND 00011100 = 00011000 = 24

Se utiliza para combinar condiciones dentro de una sentencia IF. Todas las condiciones deberán cumplirse para que se ejecute la instrucción que sigue a THEN.

(2) Un mnemónico de instrucción Z-80, que ejecuta la operación lógica AND, en forma similar a BASIC, pero tomando como operandos un dato, registro o el contenido de una posición de memoria, y el acumulador, almacenando el resultado en el registro A.

AND 30 opera el contenido de A con el valor 30, comparando bit a bit y obteniendo el valor 1 si ambos son 1 y 0 en caso contrario. El número así obtenido se almacena en A.

Permite ignorar determinados bits de un valor al operarlos con sus respectivos bits de valor 0.

Permite operar con datos, contenidos de registro o de memoria.

anidado Se dice de los bucles FOR...NEXT contenidos unos dentro de los otros. También se aplica a subrutinas utilizadas por otras subrutinas y a sentencias IF ... THEN llamadas por otras similares. En el primer caso, se precisa que los bucles se vayan cerrando de dentro a afuera, siguiendo la norma "último bucle en abrirse, primero en cerrarse".

Correcto	Incorrecto
10 FOR A=1 TO 10	10 FOR A=1 TO 10
20 FOR B=1 TO 10	20 FOR B=1 TO 10
30 PRINT A,B	30 PRINT A,B
40 NEXT B	40 NEXT A
50 NEXT A	50 NEXT B

Los bucles anidados permiten un manejo simple de las variables de conjunto, precisándose un bucle por dimensión para realizar un barrido completo de las mismas. (Veáse FOR ... NEXT, IF ... THEN y subrutinas).

archivo Una colección de datos asociada a un dispositivo. Cuando los datos son instrucciones ejecutables, se denominan archivos de programa, y son creados por BASIC u otros lenguajes de programación (PASCAL, Ensamblador, etc.). Cuando su contenido son datos, se denominan archivos de datos, así, un procesador de textos es un programa que procesa un archivo de datos, en este caso, texto. Existen distintos dispositivos que pueden soportar archivos (cassette, unidad de disco, teclado, pantalla, impresora, etc.). (Veáse dispositivo).

argumento El valor entregado a una función para su cálculo. Debe ir entre paréntesis y dependiendo de la función, será una expresión numérica, alfanumérica o lógica.

argumento fantasma Algunas funciones precisan, para su ejecución, de un argumento, si bien el resultado es independiente del valor de dicho argumento.

PRINT FRE(24)	imprime el número de bytes disponibles para programas de BASIC, independientemente de que el argumento sea 24 o cualquier otro número.

ASC Una función BASIC que entrega el código ASCII (en decimal) del primer carácter de una expresión alfanumérica. En el caso de que dicho carácter pertenezca al grupo cuyo código se compone de dos bytes, se obtiene el valor 1.

PRINT ASC("A")	imprime el número 65, código ASCII de A.

ASCII Iniciales de American Standard Code for Information Interchange. Este código asigna un byte a cada carácter, pudiéndose representar un total de 256 caracteres, incluyendo letras mayúsculas y minúsculas, signos numéricos y de puntuación y caracteres gráficos y especiales. (Véase ASC, CHR$ y juego de caracteres).

asignación Consiste en dar un valor a una variable, o bien, en guardarlo en un determinado registro o posición de memoria. Son instrucciones de asignación LET y READ.

LET A=5	asigna a A el valor 5.

ATN Una función BASIC que calcula el ángulo en radianes cuya tangente es el argumento entregado a la función. El argumento puede ser cualquier expresión numérica. Para convertir el resultado a grados sexagesimales, hay que multiplicar por 180/PI.

```
PRINT ATN(2)        imprime 1.1071487177941
                    arco tangente de 2.
```

atributo Información contenida en la memoria de video que controla el color de un punto, bloque de puntos o carácter, la posición, nombre y color de sprite. Esta información se encuentra dispuesta en forma de tablas, como la tabla de atributos de color o la tabla de atributos de sprite (figuras móviles).
Este programa altera uno de estos atributos cambiando el color de los caracteres:

```
10 CLS
20 SCREEN 1
30 COLOR 15,4,4
40 VPOKE 8200,31
50 PRINT "1234ABCD"
```

AUTO Una instrucción para utilizar en modo directo. Genera números de línea automáticamente, pudiéndose definir el principio y el incremento. Si existe una línea de programa con el número generado por AUTO aparecerá a su derecha el signo *, que deberá borrarse antes de introducir la línea. De esta forma, se evita un borrado accidental. Pulsando CTRL + STOP se detiene la generación. AUTO no puede alterar el orden de las instrucciones.
AUTO produce la serie: 10, 20, 30, ...
AUTO 100,20 produce la serie: 100, 120, 140, ...
AUTO ,5 produce la serie: 10, 15, 20 ...

bandera Un indicador que puede tomar dos estados, normalmente 0 ó 1. En BASIC se puede crear mediante una variable. En lenguaje máquina existe un registro de banderas, de 8 bits, de los cuales seis representan banderas. Se utiliza para el funcionamiento del sistema y es fundamental para la programación en este lenguaje (salto condicional). Su composición es la siguiente:

Bit 7 (S)	Bandera de signo, determina el signo de un resultado (1 negativo, 0 positivo).
Bit 6 (Z)	Bandera cero (1 si el resultado es cero).
Bit 4 (H)	Bandera de medio exceso, se se utiliza en operaciones BCD (decimal codificado en binario).
Bit 2 (P)	Bandera de paridad/sobreexceso, modifica su paridad para ciertas operaciones (lógicas, rotación ...). Adquiere el valor 1 tras una operación aritmética cuando el bit más significativo pasa de 0 á 1.
Bit 1 (N)	Bandera de suma/resta, similar a la bandera H.
Bit 0 (CY)	Bandera de exceso, contiene el bit de exceso "llevar uno" tras una operación aritmética. Las banderas son afectadas por la mayoría de las operaciones del procesador.

base Tiene dos usos posibles, como una función que toma los valores de las direcciones de memoria VRAM, donde comienzan las diversas tablas de datos

utilizadas en la visualización de los distintos modos de pantalla, o como una instrucción que asigna direcciones programadas por el usuario como comienzo de dichas tablas. Para especificar la tabla correspondiente en cada caso, se emplean los siguientes códigos:

0	Tabla de nombres de patrones en modo de texto de 40 columnas.
2	Tabla de patrones generadores en modo de texto de 40 columnas.
5	Tabla de nombres de patrones en modo de texto de 32 columnas.
6	Tabla de color en modo de texto de 32 columnas.
7	Tabla de patrones generadores en modo de texto de 32 columnas.
8	Tabla de atributos de sprite en modo de texto de 32 columnas.
9	Tabla de patrones de sprite en modo de texto de 32 columnas.
10	Tabla de nombres de patrones en modo gráfico de alta resolución.
11	Tabla de color en modo gráfico de alta resolución.
12	Tabla de patrones generadores en modo gráfico de alta resolución.
13	Tabla de atributos de sprite en modo gráfico de alta resolución.
14	Tabla de patrones de sprite en modo gráfico de alta resolución.
15	Tabla de nombres de patrones en modo multicolor.
17	Tabla de patrones generadores en modo multicolor.
18	Tabla de atributos de sprite en modo multicolor.
19	Tabla de patrones de sprite en modo multicolor.

Utilizada como función, BASE nos informa sobre la situación de las respectivas tablas de visualización. A partir de esta posición, podemos alterar determinados valores con los consiguientes cambios en la visualización. (Véase el programa ejemplo de atributo).

Utilizada como instrucción, el operador puede modificar el origen de las tablas de visualización. Esta posibilidad debe emplearse solamente cuando se conozca en profundidad el funcionamiento del VDP (Procesador de Pantalla de Video) 9129 de Texas Instruments, pues de lo contrario, la visualización en pantalla puede alterarse, o incluso, desaparecer, imposibilitando el uso del ordendor, que precisará de una reposición (RESET) con la consiguiente pérdida del contenido de memoria.

En el primer caso, el código de la tabla deberá figurar como argumento de la función, entre paréntesis.

Ejemplos:

PRINT BASE(0)	da como resultado el valor 0, origen de la tabla de nombres de patrones en modo de texto de 40 columnas.
PRINT BASE(6)	Imprime 8192, origen de la tabla de color en el modo de texto de 32 columnas.

base de datos Conjunto de datos contenidos en el ordenador que, mediante el programa o programas adecuados pueden ser modificados, ordenados

listados, ...
Por ejemplo, un inventario de los diferentes artículos de un almacén.

base de numeración El número de elementos simples que permiten expresar cantidades, sea considerados individualmente, sea combinados según la notación posicional.
La base habitual de numeración es la decimal, donde existen diez elementos simples 0-9. Dichos elementos se combinan para formar cantidades superiores a 9. Por ejemplo:

$897 = 8*10^2 + 9*10^1 + 7*10^0$

En programación se utilizan otras bases de numeración:

binaria (2)	0-1
octal (8)	0-7
hexadecimal (16)	0-9, A, B, C, D, E, F

En MSX, se pueden expresar cantidades en estas bases anteponiendo determinadas signos:

binaria &B	PRINT &B 11110000 imprime 240
octal &O	PRINT &O 7654 imprime 4012
hexadecimal &H	PRINT &H 7FA0 imprime 32672

El ordenador los reconoce y convierte a decimal.

BASIC (Beginners' All-purpose Simbolic Instruction Code). Es un lenguaje de programación interpretado de alto nivel. Fue desarrollado por Kemeny y Kurzt del Darmouth College en 1.965 para cursos de Introducción a la Informática.
No está estandarizado, existiendo numerosos dialectos, si bien los más extendidos son los desarrollados por Microsoft, entre los que se encuentra el MSX-BASIC.

baudio Velocidad de transferencia de información entre dispositivos. Equivale, aproximadamente, a 1 bit/seg.
La velocidad de transferencia de cassette en MSX puede ser de 1200 ó 2400 baudios.

BEEP Introducción que genera un pitido sonoro. Puede utilizarse en modo directo o en línea de programa, y no lleva parámetro alguno.

bifurcación La ejecución normal de un programa en BASIC se produce de forma secuencial, según números de línea crecientes. Sin embargo, este flujo puede modificarse mediante sentencias condicionales en función de los valores de una expresión para que se dirija a determinadas sentencias, creándose bifurcaciones en la ejecución del programa. (Véase IF ... THEN, ON ... GOTO).

binario Relativo al número dos. Los procesos que se desarrollan dentro del ordenador son binarios, pues tienen como origen la conducción o no conducción eléctrica a través de sus circuitos, lo que se representa por los números 1 y 0.
Estos fenómenos transcienden al operador que se ve obligado, en algunos casos, a trabajar con notación binaria para comunicarse con el ordenador, por ejemplo, en la definición de spritès.
Los procesos binarios son la base del algebra de Boole o Booleana, que se aplica a problemas de tipo Verdadero-Falso, conducción-no conducción, ...
Son binarios aquellos operadores que precisan de dos operandos:

3*2 6
4>5 Falso

BIN$ Función que ofrece, en forma alfanumérica, la expresión binaria de un argumento, pudiendo ser éste cualquier expresión numérica comprendida entre 0 y 65535. Por ejemplo:

A$ = BIN$ (255): PRINT A$ imprime 11111111

bit Unidad mínima de información. Señala uno de los dos posibles estados de un sistema binario, representándose como un 0 ó un 1.
Los bits se pueden agrupar para formar otras unidades superiores de información:

1 byte = 8 bits
1 nibble = 4 bits

BIT Un mnemónico de instrucción Z-80. Comprueba el estado 0 ó 1 de un bit de un registro del procesador (A, B, C, D, E, H ó L), o del contenido de una posición de memoria. Para ello pone la bandera Z al mismo nivel que el bit analizado.

BIT 7,A Comprueba el bit más significativo del registro A, poniendo Z al mismo nivel.

bit previo de reloj Es el bit más significativo del último byte de cada elemento de la tabla de atributos de sprite. Hace que el sprite aparezca 32 puntos a la izquierda de su posición original.

```
10 CLS
20 SCREEN 1,0
30 SPRITE$(0) = "VVVVVVVV"
40 PUT SPRITE 1,(144,92),15,0
50 A%=VPEEK(6919)
60 VPOKE 6919,A%OR128
70 PUT SPRITE 1,,15,0
80 VPOKE 6919,A%AND127
90 GOTO 40
```


BLOAD Instrucción que carga el contenido de una parte de memoria, previamente grabada con BSAVE, de especial aplicación a los programas en lenguaje máquina. Se puede especificar, opcionalmente, que se ejecute el programa inmediatamente después de cargado mediante la letra R y el incremento o decalaje para alterar el posicionamiento del programa en la memoria con una expresión numérica comprendida entre -32768 y 65535. Este incremento se añadirá a las direcciones especificadas en la grabación del programa mediante BSAVE. Ejemplo:

BLOAD "CAS:PROG 1",R,1000

Carga el programa PROG 1 del cassette 1000 posiciones de memoria por encima de su situación original en memoria, y lo ejecuta a partir de la dirección de comienzo de ejecución especificada en BSAVE.
Se aplica en forma semejante a los archivos de disco, escribiendo como nombre de dispositivo el código (A: B:) de la unidad de disco.

BSAVE Instrucción que guarda el contenido de la memoria,de especial aplicación a los programas en lenguaje máquina. Hay que especificar las direcciones inicial y final de la memoria que se quieren guardar, que serán las mismas en las que BLOAD nos cargará el programa en caso de omitirse el incremento o decalaje en dicha instrucción.
Podrá especificarse opcionalmente la dirección inicial de ejecución, en caso de utilizarse la opción R en BLOAD. Si se omite la ejecución, comenzará en la dirección inicial.

BSAVE "CAS,PROG 1",33000,35000,34000

Almacena en cassette el contenido de memoria comprendido entre las posiciones 33000 y 35000, señalando como dirección de comienzo de ejecución la 34000.

Se aplica en forma semejante a los archivos de disco, sustituyendo CAS: por A: ó B:

bucle Ejecución repetida de un cierto número de sentencias. Es uno de los elementos fundamentales de la programación.
Se pueden construir de diversas formas:

```
10 PRINT "ORDENADOR MSX"
20 GOTO 10
```

Genera un bucle indefinido. Mediante una condición IF ... THEN podemos interrumpir la ejecución del bucle:

```
10 A=1
20 PRINT "ORDENADOR MSX"
30 A=A+1
40 IF A < 20 THEN 20
```

Existe un sistema creado especialmente para la generación de bucles, la sentencia FOR ... NEXT

```
10 FOR A=1 TO 20
20 PRINT "ORDENADOR MSX"
30 NEXT A
```

Es muy compacta y potente. Los bucles no se aplican sólo en BASIC, también se dan en otros lenguajes de programación, incluso en lenguaje máquina.

buffer (1) Memoria intermedia reservada para contener un cierto tipo de información, por ejemplo, el buffer de impresora contiene los datos a enviar a la impresora. La VRAM puede considerarse como un buffer.
(2) Amplificador montado en una línea de transmisión para aumentar o adecuar la señal.

bus Conjunto de conductores eléctricos que relacionan los distintos componentes del ordenador, permitiendo su intercomunicación.

El procesador Z-80 dispone de un bus de datos de 8 conductores, que le permite transferir o recibir 1 byte simultáneamente, un bus de direcciones de 16 conductores para direccionar 64 K de RAM y un bus de control de 10 conductores para reposición, interrupción, recuperación, etc.

byte Unidad de información compuesto por 8 bits. Puede representar números naturales comprendidos entre 0 y 255, o un carácter alfanumérico (según el código ASCII).
Cada posición de memoria puede almacenar un byte.
El bus de datos puede enviar un byte simultáneamente a través de sus ocho conductores.
1 Kilobyte equivale a 1024 bytes.

cadena Serie de caracteres alfanuméricos encerrados entre comillas. Puede incluir mayúsculas, minúsculas, números, caracteres especiales y gráficos. Su longitud máxima es de 255 caracteres.
Las variables de cadena o alfanuméricos se distinguen con el signo $:

A$ = "ORDENADOR MSX"

Asigna la cadena ORDENADOR MSX a la variable alfanumérica A$. Se pueden llamar también constantes alfanuméricas.
Una cadena formada por números como "1234" no tiene valor numérico 1234, lo mismo que la cadena "TRES" no vale 3.

cadena máscara Es la cadena que sigue a una instrucción PRINT USING y altera su presentación según el formato deseado.
Los caracteres que la forman son símbolos especiales de formato. (Véase PRINT USING).

cadena nula Es una cadena que no consta de ningún carácter. Equivale al valor 0 de las constantes numéricas, siendo de importancia similar.
Se representa por dos pares de comillas seguidas, sin dejar un espacio entre ellas "" .

CALL (1) Instrucción BASIC que efectúa una llamada a una instrucción adicional contenida en un cartucho de ROM. Debe incluir el nombre de mandato y opcionalmente puede pasar parámetros al cartucho. También debe usarse para ejecutar instrucciones del sistema operativo (MSX-DOS) desde BASIC.
Puede sustituirse por el signo _ de subrayado.

CALL FORMAT A: Da formato al disco en la unidad de discos A desde BASIC.

(2) Un mnemónico de instrucción Z-80 que equivale a una llamada a subrutina. Se presenta en la forma CALL dirección o CALL condición, dirección.
La dirección de la instrucción que sigue a CALL se almacena en la pila, ejecutándose las instrucciones situadas a partir de la dirección de CALL.
Cuando se encuentra una instrucción RET (retorno) se saca la dirección almacenada en la pila y se prosigue la ejecución a partir de ésta.
Las condiciones se fijan con las banderas del registro de banderas.

canal La transmisión de información entre dispositivos exige en la mayoría de los casos que se abra un canal o camino por donde va a enviarse dicha información, lo que se hace con una instrucción OPEN. De esta forma se asigna un número a esta vía y se reserva una memoria intermedia para el trasiego de datos.

carácter Cada número, letra, signo especial o gráfico. Se identifican mediante un número, almacenado en un byte (dos en algunos casos), que es su código ASCII.
La letra A tiene el código 65, por ejemplo.

caracteres gráficos Caracteres que representan pequeños dibujos, o bloques que combinados permiten la realización de dibujos más complejos.
Los caracteres con códigos 1-31 y 192-223 son, en su mayoría, de este tipo.
El primer grupo precisa para su representación de 2 bytes.

Si se utiliza la función CHR$ el primer código debe ser 1, y al segundo se le debe sumar el valor 64:

```
PRINT CHR$(1) + CHR$(64+5)
```

Imprime un trébol (código 5).
Los caracteres gráficos ocupan una matriz de puntos de 8x8, por lo que se pierde una parte al visualizarlos en el modo de texto de 40 columnas, que utiliza una matriz de 6x8.

```
SCREEN 1:PRINT CHR$(215)
```

Imprime un carácter ajedrezado de 4x4 cuadros.

```
SCREEN 0: PRINT CHR$(215)
```

Lo reduce a 3x4 cuadros.

cargador de código máquina Un programa que almacena en memoria el código máquina, representado normalmente en hexadecimal. En este caso, se llama también cargador hexadecimal.
Es útil en programas cortos, porque si no la búsqueda de códigos se hace difícil, siendo necesario recurrir a un ensamblador.

cargador hexadecimal (Véase Cargador de Código Máquina). El siguiente programa solicita una dirección de comienzo y el código máquina en hexadecimal, cargándolo en memoria a partir de dicha dirección.

```
10 CLEAR 200,55000!:CLS
20 LOCATE 0,0:INPUT " INTRODUCIR DIRECCION DE
   COMIENZO  ";CM
30 IF CM<55000! OR CM>62336! THEN GOTO 10
40 A$=""
50 LOCATE 10,23:INPUT A$
60 IF A$="T" OR A$="t" THEN STOP
70 LOCATE 10,21:PRINT CM;" ";A$
80 POKE CM,VAL("&H"+A$)
90 CM=CM+1
100 GOTO 40
```

Asímismo, el programa reserva el espacio para el código máquina desde la dirección 55000 mediante CLEAR

cassette Dispositivo secuencial basado en soporte magnético, para Entrada y Salida, de programas y datos. Su utilización está ampliamente implementada en MSX-BASIC (Instrucciones SAVE, LOAD, CSAVE, CLOAD, CLOAD?, BSAVE, BLOAD, OPEN, PRINT#, INPUT#, CLOSE, MERGE, MOTOR, ...).
Su velocidad de transferencia puede fijarse en 1200 ó 2400 baudios. Su nombre de dispositivo es CAS: Soporta archivos secuenciales.
Es lento comparado con los discos magnéticos y carece de la flexibilidad de su utilización, sin embargo, es el dispositivo más empleado en ordenadores domésticos.

CCF Un mnemónico de instrucción Z-80. Complementa la bandera de exceso, es decir, cambia su estado de 0 a 1, ó de 1 a 0.

CDBL Función numérica que convierte el resultado de cualquier expresión numérica a doble precisión.
Este cambio se realiza a efectos internos del ordenador.

choque de sprites Suceso producido por el solapamiento de zonas no transparentes de al menos dos sprites. Se detecta mediante las instrucciones ON SPRITE GOSUB y SPRITE ON, OFF, STOP.
Si bien no pueden conocerse los sprites afectados ni su posición.
Tampoco pueden detectarse choques entre sprites y fondo.

CHR$ Función que toma como argumento un código ASCII y nos ofrece el carácter correspondiente.

```
PRINT CHR$(77) + CHR$(83) + CHR$(88)
```

Imprime MSX.
Los caracteres con códigos comprendidos entre 1 y 31 precisan de dos códigos para su presentación.

```
PRINT CHR$(1) + CHR$(64+5)
```

Imprime un trébol.
La función inversa de CHR$ es ASC.

CINT Función que convierte el resultado de una expresión numérica a precisión entera.

```
A=CINT(3.141592):PRINT A
```

Imprime el valor 3.
Dado que la precisión entera no admite parte fraccionaria, ésta es eliminada (no se redondea).

CIRCLE Instrucción que traza una circunferencia, elipse o arcos de ambos en las dos pantallas gráficas (Alta resolución y multicolor).

Los parámetros de esta instrucción son:

Coordenadas del centro:	van entre paréntesis. Pueden ser absolutas o relativas (STEP).
Radio:	medido en número de puntos (pantalla alta resolución).
Color:	según los códigos habituales. Es opcional
Angulo inicial:	en radianes. Si es negativo traza la línea de cierre del sector angular. Es opcional.

Angulo final: en radianes. Si es negativo traza la línea de cierre del sector angular. Es opcional.

Altura/anchura

Esta relación entre altura y anchura fija el aspecto del círculo, toma por omisión el valor 1 que produce "círculos" algo achatados. La relación 1.2 produce círculos más reales.
Si la relación es mayor que 1, se mantiene el radio vertical disminuyendo el horizontal.
Si la relación es menor que 1, se mantiene el radio horizontal.
Los círculos tienen como origen el valor 0, zona positiva del eje X, y como fin 2*PI, en sentido antihorario. Ejemplo:

```
10 SCREEN 2
20 CIRCLE(128,96),90,15,-3.141592/2,-1E-10,65
30 GOTO 30
```

Este programa imprime un sector circular comprendido entre 90 y 360 grados sexagesimales.

circuito Un conjunto de componentes electrónicos interconectados, capaces de realizar una función determinada.
Se distinguen:
los circuitos cableados a mano; donde las conexiones son cables soldados uno a uno.
Los circuitos impresos; placas aislantes con pistas conductoras impresas sobre las que se colocan los componentes, soldándolos simultáneamente.
Los circuitos integrados; base de la revolución informática, constituidos por un chip (pedacito) de silicio sobre los que se "crean" los distintos componentes por procedimientos fotográficos y químicos.

Los más evolucionados son los circuitos LSI de integración a gran escala, que en su mayoría conforman los ordenadores MSX. Entre ellos se encuentran el propio Microprocesador Z-80 A, el Generador Programable de Sonido (PSG) AY-8910, el Procesador de Pantalla de Video (VDP) 9129 y el Interfaz Programable para Periféricos (PPI) 8255.

CLEAR Una instrucción BASIC que asigna el valor 0 a las variables numéricas, la cadena nula a las variables alfanuméricas, desdimensiona las matrices, cierra los archivos y, opcionalmente, permite alterar el espacio reservado en memoria para el tratamiento de cadenas (200 bytes por omisión) y la posición superior de memoria. También se pierde el retorno de las sentencias que contienen GOSUB.
Se utiliza mucho en modo directo para depuración.

CLEAR 200	Borra las variables anteriormente asignadas.
CLEAR 250	Además asigna un espacio de 250 bytes para cadenas. Se utiliza al aparecer el mensaje de error "out of string space".
CLEAR ,40000	Reserva los bytes comprendidos entre 40000 y 62336 de forma que BASIC no puede invadir dicho espacio.

Esta última técnica es la empleada con el fin de reservar memoria para programas en código máquina.

CLOAD Instrucción BASIC para cargar del cassette un programa en BASIC previamente grabado con CSAVE.

CLOAD "PROG 1"   Carga el programa PROG 1 del cassette.

Si omitimos el nombre del programa, cargará el primero que encuentre.
Puede añadirse el nombre de dispositivo:

CLOAD "CAS: PROG 1"   Semejante al anterior.

Antes de cargar el programa, CLOAD cierra los archivos y anula las variables.

CLOAD? Instrucción que verifica si la grabación de un programa mediante la sentencia CSAVE ha sido correcta. Para ello realiza una comparación entre el programa en memoria con el programa que se recibe del cassette.
Si la comprobación no es correcta, aparece el mensaje "Verify error", y OK en caso contrario.
CLOAD? comprueba si el siguiente programa de la cinta coincide con el de la memoria.

CLOAD? "PROG 1"

Lee la cinta hasta encontrar el programa PROG 1, comparándolo con el existente en la memoria.

CLOSE Instrucción para cerrar los archivos.

CLOSE #1,#2   Cierra los archivos #1, #2

Si no especificamos ningún archivo, cierra todos los que estén abiertos.
La función inversa, que abre los archivos, es OPEN
La instrucción CLOSE cancela las memorias intermedias asociadas a los archivos. Envía el contenido de las memorias intermedias de salida.

CLS Instrucción para borrar la pantalla.
En las pantallas de gráficos el cambio de color de fondo especificado mediante una instrucción COLOR, no se ejecutará hasta encontrarse con un CLS.

En los modos de texto, el cursor se desplaza a la esquina superior izquierda.

código máquina Es el código por el que se representan las instrucciones del lenguaje de máquina, es decir, el lenguaje entendido por el microprocesador.
Está compuesto por números, generalmente en base hexadecimal, (aunque en principio podrían escribirse en binario o decimal) por ser más compactos.
Es extraordinariamente rápido, al no precisar interpretación, aunque está sujeto a errores

(Véase lenguaje de máquina).

COLOR (1) Instrucción para determinar los colores de las distintas zonas de pantalla (primer plano, fondo y bordes).
Cada color se especificará según el código de la siguiente tabla:

0	Transparente	1	Negro
2	Verde	3	Verde claro
4	Azul oscuro	5	Azul claro
6	Rojo oscuro	7	Azul celeste
8	Rojo	9	Rojo claro
10	Amarillo oscuro	11	Amarillo claro
12	Verde oscuro	13	Magenta
14	Gris	15	Blanco

Ejemplo:

COLOR 1,15,2   El primer plano adquiere el color negro, el fondo se hace negro y los bordes verdes.

También se admiten las siguientes variantes:

COLOR 1 COLOR 15,1
COLOR,1 COLOR 15,,1
COLOR,,1 COLOR,15,2

Se puede utilizar en todos los modos. En el modo de texto de 40 columnas, el color de borde es el de fondo. En ambos modos gráficos para que se realice el cambio de color es necesaria la utilización de un CLS.
La tabla de color de la VRAM es susceptible de modificación en todos los modos.
(2) Los ordenadores MSX permiten la representación de hasta 16 colores diferentes. Sin embargo, existe una limitación en el uso de color en las líneas horizontales, que en grupos de 8 puntos solamente admiten dos colores como máximo "manchando" de uno de ellos los puntos del grupo, en caso de que se intente usar colores diferentes.
El nuevo standard MSX2 permite hasta 256 colores.

columnas reservadas En los modos de texto de 40 y 32 columnas, se reservan las dos columnas situadas a la izquierda y una de la derecha; no se emplean, si bien puede accederse a ellas a través de la VRAM.

comillas Un signo especial que se emplea en MSX-BASIC como delimitador de cadenas alfanuméricas.

"ORDENADOR MSX" Es considerado por el ordenador como una única cadena alfanumércia, incluyendo el espacio intermedio.

compilador Un programa que parte de un programa fuente, es decir, del texto de una serie de instrucciones y lo traduce a lenguaje máquina, creando un programa objeto.
Este programa, junto con un editor, constituyen un lenguaje de programación.
El proceso descrito se denomina compilación, y es alternativo al proceso de interpretación, que en vez de traducir de una sola vez, y para siempre, el programa fuente a objeto, lo traduce instrucción a instrucción y solamente para ejecutarlo en ese momento.

complemento de dos (Véase Notación de Complemento de Dos).

comprobación Consiste en analizar un programa para ver si se ejecuta adecuadamente, al haberse eliminado los errores de programación, y si cumple las tareas para las que fue creado.
Para ello se pone en marcha el programa para un supuesto resuelto a mano, comprobando los resultados.
También se hace funcionar con datos límite para ver en qué margen de datos opera adecuadamente.

concatenación Operación por la que se unen dos cadenas alfanuméricas formando una única cadena. Se realiza con el operador + :

PRINT "ORDEN" + "ADOR"

Imprime ORDENADOR.

A$ = "ORDEN" + "ADOR"

Asigna a A$ la cadena ORDENADOR.

condición Situación o circunstancia que debe cumplirse para que se realice una acción determinada.

Permite al ordenador realizar acciones diferentes según los resultados obtenidos. Por ejemplo, generar un ruido de explosión al producirse un choque de sprites.
En BASIC se implementan mediante la instrucción IF ... THEN, aunque también pueden incluirse en el grupo las instrucciones ON ... GOTO, ON SPRITE GOSUB y similares.
En lenguaje máquina se concreta en saltos (JP, JR) o llamadas o subrutinas (CALL) condicionales en función del estado de las banderas de paridad, cero, exceso, etc.

condicional Se dice de una instrucción que incluye una condición, y cuya ejecución se ve modificada por ésta.

```
10 IF A<>0 THEN 100 ELSE 200
```

Si A=0 la ejecución del programa seguirá en la línea 200, si no, en la 100.

```
JR NZ,desplazamiento
```

Salta a una posición determinada por el valor de desplazamiento, siempre que A no haya alcanzado el valor cero.

conector Elemento compuesto por dos piezas, una que se monta en el exterior de los distintos dispositivos del ordenador y otra que se suelda a los extremos de un cable.
Ambas se conectan entre sí, produciéndose un contacto entre los conductores del cable y los circuitos internos de los dispositivos, permitiendo su comunicación.

Cassette:	Conector DIN 8 patillas
Impresora:	Conector tipo Centronics Amphenol 14 patillas.
Mando de juegos:	Conector AMP 9 patillas.

configuración MSX La configuración del sistema MSX incluye un microprocesador Z-80 A, un procesador de pantalla de video (VDP), un generador de sonido programable (PSG) y un interfaz programable para periféricos (PPI). (Véase circuito)
El lenguaje de la configuración es el MSX-BASIC de Microsoft versión 1.0 (1.983) y el sistema operativo que le acompaña, contenidos en 32K de ROM.
La memoria RAM mínima es de 8K, si bien hasta la fecha en Europa se han superado los 16K.
Existe una memoria RAM para video (VRAM) de 16K.

conjunto Un grupo de elementos, generalmente homogéneos.
En BASIC se puede representar conjuntos de elementos homogéneos con un único nombre de variable, al que se añade unos subíndices para representar cada elemento concreto.
Para ello, debe dimensionarse el conjunto.

```
DIM A! (19,19)
```

Dimensiona una tabla de elementos de simple precisión de 20x20.

```
A! (0,0)
```

Representa el primer elemento de la tabla. El valor 0 es admitido como un valor más.
La potencia de las variables de conjunto reside en su capacidad de ser "barridas" mediante la variación de índices generados por bucles del tipo FOR ... NEXT.

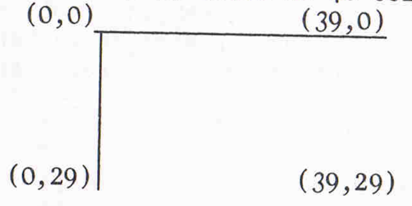
CONT Instrucción que continua la ejecución de un programa tras haber sido éste detenido por un CTRL+STOP o por un STOP del programa.
El flujo se reanuda en la instrucción siguiente a la detención, salvo en el caso de que ésta halla ocurrido en un INPUT, en que volverá al INPUT.

En caso de editar una sentencia se perderá la facultad de continuar.

contador de programa Un registro del procesador Z-80. Se representa por PC y consta de 16 bits. Contiene la dirección de la instrucción a ejecutar a continuación.

coordenadas Par de valores numéricos que representan un punto del plano con relación al origen de unos ejes cartesianos.
En BASIC, la disposición de ejes y coordenadas es la siguiente:

Pantalla de texto de 40 columnas



Pantalla de texto de 32 columnas

(0,0) (31,0)

(0,23) (31,23)

En las pantallas de texto, las coordenadas representan la posición del cursor de texto. El cursor señala la posición en que se imprimirá a continuación, y su situación se determina mediante la instrucción LOCATE.

Pantalla de gráficos de alta resolución y multicolor

(0,0) (255,0)

(0,192) (192,255)

Las coordenadas así definidas se denominan absolutas y se representan entre paréntesis, separadas por una coma en la mayoría de las instrucciones gráficas.
Las coordenadas relativas también representan la posición de un punto pero con relación al último punto referenciado (cursor de gráficos). Se representan entre paréntesis y separadas por comas, con la clave STEP delante.

```
10 SCREEN 2
20 COLOR 15,4,4
30 PSET (128,96)
40 PSET STEP (5,5)
```

La instrucción 30 ilumina de color de primer plano (blanco=15) un punto de coordenadas absolutas (128,96) que corresponden al centro de la pantalla. La instrucción 40 ilumina del mismo color un punto situado a 5 unidades hacia la derecha y hacia abajo del punto anterior, y que tendrá por consiguiente, las coordenadas absolutas (133,101).

COPY Instrucción del MSX-DOS residente en disco. Copia el archivo especificado en primer lugar con el nombre y en la unidad de disco especificado en segundo lugar.

Si no se especifica unidad de disco, se asume la unidad por omisión. Si no se especifica nombre en segundo lugar, se copia con el mismo nombre.

```
COPY A:PROG1.BAS B:PROG2.BAS
```

Copia el programa BASIC PROG1 de la unidad A en la unidad B con el nombre PROG2.

```
COPY A:PROG1.BAS B:
```

Copia PROG1 de A en B con el mismo nombre.

```
COPY PROG1.BAS PROG2
```

Copia PROG1.BAS en la misma unidad con el nombre PROG2.BAS

COS Función que ofrece el coseno del ángulo dado en radianes.

```
A=COS(3.141592): PRINT A
```

Imprime -.99999999999978

CP Mnemónico de instrucción Z-80 que realiza una comparación entre el contenido del registro A y los restantes registros, contenidos de memoria o datos.
Si ambos coinciden Z toma el valor 1, si no, toma el valor 0.
Si el número a comparar es menor que A, la bandera de exceso C toma el valor 0, y 1 en caso contrario.

CPD Mnemónico de instrucción Z-80 que compara el valor de A con el contenido de la posición de memoria (HL), si coinciden Z toma el valor 1, y 0 en caso contrario. A continuación HL y BC disminuyen en una unidad.
El contenido de BC representa la amplitud del intervalo explorado y HL la posición de comienzo de la exploración.

CPDR Mnemónico de instrucción Z-80, similar a CPD, pero que repite la comparación siempre que no se dé alguna de las siguientes condiciones:

- Los valores a comparar sean iguales.
- BC haya alcanzado el valor 0.

Permite explorar un bloque de memoria, buscando un valor semejante al del acumulador, desde una posición HL hacia posiciones inferiores.

CPI Mnemónico de instrucción Z-80, similar a CPD, pero que incrementa el registro HL en una unidad, y disminuye el BC también en una unidad.

CPIR Mnemónico de instrucción Z-80, similar a CPDR pero que incrementa HL en una unidad.
Permite explorar un bloque de memoria, buscando un valor semejante al del acumulador, desde una posición HL hacia posiciones superiores.

CPL Mnemónico de instrucción Z-80 que altera el estado de cada uno de los bits del acumulador. Si A toma el valor 11001100, al ejecutar CPL tomará el valor 00110011.

CPU (Véase Unidad Central de Proceso).

CSAVE Introducción para almacenar un programa BASIC en cassette, en formato abreviado. Los programas grabados con CSAVE no se pueden refundir, pues para ello deben grabarse en formato ASCII.

```
CSAVE "CAS:PROG1"
```

El nombre del programa del programa constará como máximo de seis caracteres, y el primero no podrá ser un carácter numérico; los que se especifiquen en exceso, no se tendrán en cuenta.
CSAVE se puede utilizar con dos velocidades de

grabación: 1200 baudios (es la usual y la empleada en caso de omisión) y 2400 baudios.

```
CSAVE "nombre",1    Se selecciona 1200 baudios
CSAVE "nombre",2    Se selecciona 2400 baudios
```

CSNG Función que convierte datos numéricos en datos de simple precisión.

```
PRINT 1/3           Imprime .33333333333333
PRINT CSNG(1/3)     Imprime .333333
```

CSRLIN Función que nos indica la coordenada según el eje Y de la posición del cursor de texto.

```
A=CSRLIN: PRINT "fila"; A
```

cursor En las pantallas de texto es un rectángulo brillante (del color de primer plano) que señala la siguiente posición de impresión.
El cursor se puede hacer desaparecer mediante la instrucción LOCATE.

```
LOCATE,,0          Apaga el cursor
LOCATE,,1          Enciende el cursor
```

En las pantallas gráficas, el cursor de gráficos no se materializa en la pantalla, pero queda almacenado en memoria (sus coordenadas) y sirve como punto de referencia para las coordenadas relativas.

CVD Función de MSX-BASIC de disco que "recompone" los valores numéricos de doble precisión que habían sido transformados en cadenas alfanuméricas para su almacenamiento en archivos aleatorios.
Es la función inversa de MKD$.
Toma como argumento una cadena de 8 bytes, obtenida al aplicar MKD$ a una valor numérico de doble precisión.


```
PRINT CVD ("cadena de 8 bytes")
```

Imprime el valor de doble precisión que corresponde a la cadena de 8 bytes.

CVS Función de MSX-BASIC de disco, semejante a CVD pero en precisión simple.
Es la función inversa de MKS$. Su argumento es una cadena de 4 bytes de longitud.

```
PRINT CVS ("cadena de 4 bytes")
```

Imprime el valor de precisión simple que corresponde a la cadena de 4 bytes.

CVI Función de MSX-BASIC de disco, semejante a CVD pero en precisión entera.
Es la función inversa de MKI$. Su argumento es una cadena de 2 bytes de longitud.

```
PRINT CVI ("cadena de dos bytes")
```

Imprime el valor de precisión entera que corresponde a la cadena de dos bytes.

DAA Un mnemónico de instrucción Z-80 que convierte el contenido del registro A a notación BCD (Decimal Codificado en Binario).
Se utiliza a continuación de operaciones matemáticas (suma y resta).

DATA Sirve para el almacenamiento de constantes numéricas o alfanuméricas, a las que se accede a través de la instrucción READ.
Los datos numéricos y de cadena han de ir separados por comas.

```
DATA 5,ABC,63.5,A3D
```

Cuando los datos alfanuméricos contengan comas, dos puntos o espacios al principio o final, estos deberán ir entre comillas.

```
DATA 5,"AB,C",63.5," A3D"
```

(Véase READ, RESTORE).

DATE Instrucción del Sistema Operativo de disco MSX-DOS que presenta y pide actualización de fecha. Se utiliza en el almacenamiento de archivos de disco, para identificar la fecha en que fueron almacenados.
La fecha se introduce mediante dos dígitos de mes, dos de día y hasta cuatro para el año, separados por / .

datos Información en cualquiera de sus formas. Se suele distinguir entre datos y resultados. Los datos son información inicial, de partida, para la resolución de un problema concreto. Los resultados son la información obtenida con su resolución.
El MSX-BASIC distingue datos numéricos, alfanuméricos y lógicos.

DEC Un mnemónico de instrucción Z-80 que disminuye en una unidad el valor de un registro, par de registros o posición de memoria. Si dicho valor, como consecuencia de esta disminución, alcanza el valor 0, la bandera Z tomará el valor 1.
Algunas instrucciones realizan un decremento automático como CPD o CPDR.

DEFB Una instrucción de ensamblador. Se introduce entre los mnemónicos editados en el ensamblador para indicar a éste que coloque el valor de las sucesivas expresiones que siguen a DEFB en las posiciones de memoria siguientes a la última instrucción ensamblada.
Por lo tanto, el ensamblador calcula el valor literal de las expresiones (deben estar comprendidos entre 0 y 255) y lo coloca en memoria.
Se emplea en la generación de tablas o para reservar espacio de trabajo.

```
.DEFB 0,255,255,24,24,24,24,0
```

Asigna los valores 0, 255, 255, 24 ... a las sucesivas posiciones de memoria, si estos valores se trasladan, por ejemplo, al área de definición de un sprite de 8x8, se obtiene una T.

DEFDBL Semejante a DEFINT pero para definir variables de doble precisión.

```
DEFDBL A,C-F
```

Declara de doble precisión todas las variables que comienzan por A, C, D, E o F.

DEF FN Instrucción para definir una función de hasta nueve variables.
La función se define añadiendo un nombre de variable inmediatamente después de DEF FN. A continuación, entre paréntesis, se expresan unos parámetros que se igualan a una relación funcional que los contiene.
La función se utiliza escribiendo FN e inmediatamente después el nombre de variable con que se definió. A continuación, entre paréntesis, se expresan los valores que se van a dar a los parámetros.
Cuando la ejecución del programa alcanza una FN, el ordenador busca la definición correspondiente DEF FN y sustituye los nuevos valores en el lugar de la relación funcional allí definida calculando el valor de la función, que asigna a FN, continuando la ejecución normal del programa.

```
10 DEF FNA(X,Y)=(2*X-Y^2)/X
20 B=FNA(2,3)
30 PRINT B
RUN
```

Imprime el valor -2.5
La línea 10 define una función FNA que expresa una relación entre los parámetros X,Y dada por:

(2*X-Y^2)/X

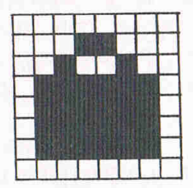
La línea 20 utiliza la función FNA sustituyendo X por 2 e Y por 3, si bien podría haberse utilizado cualquier otra expresión numérica.

definición de sprite Los sprites (figuras móviles) en el sistema MSX pueden ser de dos tipos: de 8x8 ó de 16x16 puntos.
Su definición se realiza mediante la instrucción:

SPRITE$ (nº de sprite) = cadena de definición de sprite.

El número de sprite debe estar comprendido entre 0-63 para 16x16 puntos y 0-255 para 8x8 puntos.
La cadena de definición de sprite está formada por caracteres cuyo código ASCII sea el valor de un patrón de ocho puntos que forme una línea (para 8x8) o media línea (para 16x16) del sprite.

Supongamos que queremos crear un sprite de 8x8 como el de la figura:



Si utilizamos la notación binaria, formamos cada línea mediante el prefijo &B que nos dá el valor decimal del patrón de puntos (0 en blanco, 1 en negro). En nuestro ejemplo sería:

```
CHR$ (&B00000000)
CHR$ (&B00011000)
CHR$ (&B00100100)
CHR$ (&B01111110)
CHR$ (&B01111110)
CHR$ (&B01111110)
CHR$ (&B01111110)
CHR$ (&B00000000)
```

Lo que equivale a:

```
CHR$(0) + CHR$(24) + CHR$(36) + CHR$(126) +
CHR$(126) + CHR$(126) + CHR$(126) + CHR$(0)
```

Que sería la cadena de definición.
Para sprites de 16x16 puntos se sigue el mismo procedimiento, formando cuatro subcadenas de una de las esquinas de 8x8 en el siguiente orden:

esquina superior izquierda + esquina inferior izquierda + esquina superior derecha + esquina inferior izquierda.

Estas cuatro subcadenas se unen y se asignan a la definición.

DEFINT Instrucción que declara una variable numérica o conjunto de ellas como variable de precisión entera.


```
DEFINT M,R
```

Define a todas las variables que comienzan por M o R como variables de precisión entera.

```
DEFINT B-D
```

Define a todas las variables que comienzan por B, C o D de tipo entero.
Esta asignación de tipo puede evitarse para una variable aislada mediante un carácter de declaración de tipo añadido al nombre de la variable con posterioridad a la utilización de DEFINT.

DEFM Una instrucción de ensamblador. Indica al ensamblador que coloque los códigos ASCII de los caracteres que componen la cadena que sigue a DEFM en las posiciones de memoria que siguen a la última instrucción ensamblada.

```
DEFM "ABCD"
```

Asigna los valores 65, 66, 67 y 68 a las siguientes posiciones de memoria.
Permite crear tablas con mensajes o texto.

DEFS Una instrucción de ensamblador. Reserva el número de posiciones de memoria dado por una expresión numérica, y lo rellena con ceros.

```
DEFS 200
```

Reserva 200 bytes de memoria a los que asigna el valor 0.
Se utiliza para reservar espacio de trabajo en memoria.

DEFSNG Semejante a DEFINT pero para definir variables de precisión simple.

```
DEFSNG H,J-L
```

Declara como variables de precisión simple las que empiezan por H, J, K o L.

DEFSTR Semejante a DEFINT pero para definir variables de tipo alfanumérico.

DEFSTR M,O-Q

Declara como variables alfanuméricas las que empiezan por M, O, P o Q.

DEFUSR Una instrucción BASIC que especifica una dirección de la memoria.
Se utiliza, en unión de USR, para ejecutar un programa o rutina en código máquina cuya dirección de comienzo de ejecución sea la especificada por DEFUSR.
Permite, mediante un parámetro comprendido entre 0 y 9, prefijar hasta diez direcciones de entrada.
Admite como dirección cualquier expresión numérica en el intervalo 0-65535.

```
DEFUSR1=&HOOCO
DEFUSR2=&HOOC3
X=USR1(R)
X=USR2(R)
```

En este ejemplo se prefijan dos direcciones de rutinas en código máquina del BIOS. La situada en HOOCO produce en "bip" y la de HCOOC3 borra la pantalla (equivalente a CLS).

DEFW Una instrucción de ensamblador semejante a DEFB pero con expresiones comprendidas entre 0 y 65535, que coloca en dos posiciones de memoria (2 bytes) con el byte menos significativo en primer lugar.

DEFW 40000

Situa el valor 64 en la posición de primera memoria y el 156 en la segunda, a partir de la última instrucción ensamblada.

((156x256)+64)=40000

DELETE (1) Instrucción BASIC para borrar de la memoria una o varias líneas de programa.
Si pretendiéramos borrar una sola línea de programa, es suficiente con teclear el número de ésta y RETURN.

DELETE 10	Borra la línea 10.
DELETE 10-30	Borra de la línea 10 a la 30, ambas inclusive.
DELETE -20	Borra desde la primera línea del programa hasta la 20.
DELETE 20-.	Borra desde la línea 20 hasta la última editada.

Si lo usamos sin parámetros, borrará la última línea visualizada por LIST o por cualquier error.

(2) Instrucción del sistema operativo de disco MSX-DOS que elimina un archivo de un disco. Es semejante a la instrucción KILL de BASIC de disco.
Para borrar el archivo debe especificarse su nombre y, opcionalmente, la unidad donde se encuentra

DELETE A: PROG1.BAS

Borra el programa PROG1 en BASIC que se encuentra en la unidad A.

depuración Consiste en la eliminación de errores en programas o datos.
(Véase comprobación).
En programas que requieran la instrucción de numerosos datos, es necesario incluir unas rutinas que permitan visualizar los datos para comprobarlos y sustituirlos en caso de error.

desensamblador Un programa que traduce el código máquina a los mnemónicos de instrucciones del procesador.
Permite ver las instrucciones de que se compone un determinado programa en código máquina.

detección de interrupciones El sistema MSX dispone de una serie de instrucciones que permiten la detección de interrupciones. Las interrupciones pueden generarse a partir de:

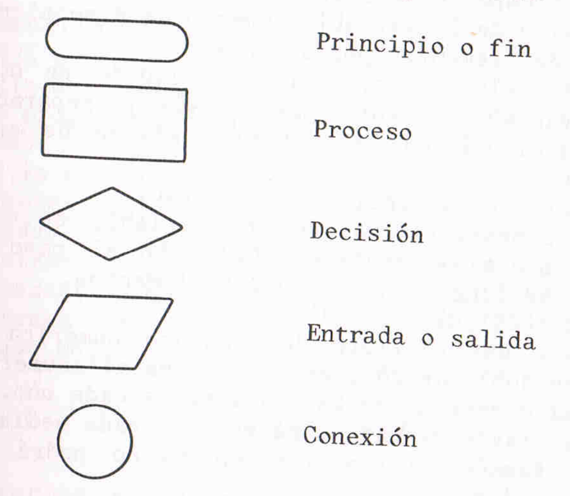
- La aparición de un error durante la ejecución del programa ON ERROR GOTO.
- El transcurso de un determinado período de tiempo ON INTERVAL GOSUB.
- La pulsación de una tecla de función ON KEY GOSUB.
- Un choque entre sprites ON SPRITE GOSUB.
- La pulsación de las teclas CTRL+STOP. ON STOP GOSUB.
- La pulsación de barra de espacio o disparadores de los mandos de juego ON STRIG GOSUB.

Estas instrucciones dirijen la ejecución del programa a una rutina de servicio de la interrupción.
Para que actúen, hay que activar el proceso de interrupción, para ello se emplean: KEY ... ON, STOP ON, SPRITE ON, INTERVAL ON.
Para desactivarlas se sustituye ON por OFF.
Existe la posibilidad de suspender la detección de interrupciones, sustituyendo ON por STOP. De esta forma, se memoriza si se ha producido la acción generadora de interrupción y al activarse de nuevo, se ejecuta la rutina de servicio.
(Véase Interrupción).

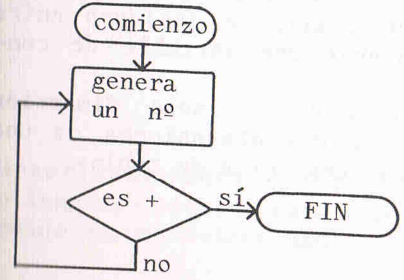
DI Un mnemónico de instrucción Z-80. Desactiva las interrupciones del procesador.

El procesador puede recibir señales de interrupción exteriores a través de INT (línea de control del procesador). Dichas interrupciones se llaman enmascarables, pues pueden desactivarse con DI.

diagrama de flujo Permiten representar esquemáticamente los distintos pasos de un algoritmo. Existen unos signos convencionales de las distintas acciones. Los principales son:



Los respectivos pasos se representan mediante trazos.



En el ejemplo se genera un número, si es negativo se detiene la ejecución, si es positivo se vuelve a generar otro, y así, sucesivamente.

DIM Instrucción BASIC que crea variables de conjunto, reservando un área en memoria para su almacenamiento y permitiendo su reconocimiento como tales por el ordenador.
Al mismo tiempo, especifica el tipo y en su caso, la precisión de la variable, número de dimensiones y número de elementos por dimensión.
Para utilizarla se escribe DIM seguido de un nombre de variable y entre paréntesis y separados por comas el número de elementos máximo de cada dimensión.
El elemento con subíndice 0 es válido.
Se puede dimensionar más de una variable de conjunto en una misma sentencia DIM. En el caso de cadenas, su longitud será de 255 elementos.

```
DIM A(25),B$(15,2,3)
```

Dimensiona una variable de conjunto numérica de precisión doble de 26 elementos y una alfanumérica de 16x3x4 elementos de 255 caracteres cada uno.
Si una variable de matriz no es declarada mediante DIM, el tamaño de sus dimensiones no podrá ser superior a diez.

dimensión Cada una de las expresiones numéricas que, separadas por comas, se incluyen entre paréntesis para dimensionar una variable de conjunto.
Una variable de conjunto de una sola dimensión es una lista (vector), de dos dimensiones es una tabla (matriz), de tres, una serie de tablas.

DIR Instrucción del sistema operativo de disco. Visualiza la relación de archivos (Directorio) que se encuentran en la unidad especificada o, en su defecto, en la unidad asumida por omisión.
Se incluyen, asímismo, la longitud de archivo y la fecha de su creación.

```
DIR A:
```

Visualiza los archivos de la unidad A.

dirección Número entero comprendido entre 0 y 65535 que especifica una de las posiciones de memoria del ordenador.
Los programas en BASIC comienzan en la dirección 32768.
Por la especial notación del sistema MSX, las posiciones superiores a 32767, se muestran en determinadas ocasiones con signos negativos, y en orden creciente a partir de la posición 32768 que se representa como -32768 hasta la 65535 que se representa como -1.
(Véase Notación de complemetno de dos).

0		65535
0	32767 | -32768	-1

dirección absoluta El valor por el que el ordenador reconoce una de las posiciones de memoria. (Véase dirección).

direccionamiento Procedimientos utilizados en lenguaje máquina para especificar un dato contenido en una determinada dirección de memoria.

El procesador Z-80 admite direccionamiento en diez formatos distintos.

1 Direccionamiento implícito: no se precisa dirección ni dato.
NEG,RET

2 Direccionamiento inmediato: la dirección que que contiene al dato es la siguiente a la de la instrucción.
LD A,5

3 Direccionamiento inmediato extendido: el dato se encuentra en las dos direcciones que siguen a la instrucción.
LD HL,30000

4 Direccionamiento relativo: se suma un valor con signo (-128 a 127) a la dirección del contador de programa, obteniéndose la nueva dirección de ejecución.
JR NZ 120

5 Direccionamiento por registro: el dato se encuentra en el registro incluido en la instrucción.
LD A,B

6 Direccionamiento por registro indirecto: la dirección del dato se encuentra en un par de registros.
LD A,(HL)

7 Direccionamiento extendido o direccionamiento directo: la dirección del dato se encuentra en las dos direcciones que siguen a la instrucción.
LD A,(30000)

8 Direccionamiento indicado (indexado): la dirección se obtiene sumando un desplazamiento a la dirección contenida en el índice.
LD A,(IY+3)

9 Direccionamiento de página cero: utilizada por las instrucciones RST, que equivalen a una llamada a subrutina en las posiciones 0-56 en grupos de ocho.

10 Direccionamiento por bit: no direcciona bytes , sino bits.
RES 5,A

disco Un dispositivo para el almacenamiento de información, que se caracteriza por su alta velocidad de acceso y transferencia.
Es un disco de plástico cubierto de material magnético. Su manejo y control es realizado por una parte especializada del D.O.S. (Sistema Operativo de Disco) por lo que el usuario no tiene que preocuparse de donde están los archivos, o de buscar los "huecos" para almacenarlos.
El disco admite el tratamiento de archivos aleatorios. El nombre de dispositivo depende de la unidad y se expresa por una letra seguida de dos puntos.

A: Representa la unidad de disco A.

La unidad de disco que contiene el sistema operativo MSX-DOS en el momento del arranque o reposición del sistema es la unidad asumida por omisión, que actúa cuando no se especifica unidad alguna.

DISKCOPY Instrucción del sistema operativo de disco (MSX-DOS) que permite obtener una copia de un disco.
Si se dispone de dos unidades de disco el proceso se realiza automáticamente.

DISKCOPY B: A:

Copia el contenido del disco que se encuentra en B en el disco que hemos introducido en A.
Si sólo hay una lectora, aparecen mensajes en pantalla con instrucciones para la copia.

dispositivo Aparato físico asociado al ordenador, que soporta archivos.
Se reconocen por su nombre clave, que suele incluirse como parámetro en las instrucciones.

cassette	CAS:
pantalla de texto	CRT:
pantalla de gráficos	GRP:
impresora	LPT:
cartucho	CAT:
unidad de disco A	A:
unidad de disco B	B:

Algunos dispositivos soportan varios archivos, por lo que para hacer referencia a uno concreto, hay que emplear su nombre completo (cassette, unidades de disco).
Otros dispositivos sólo soportan simúltaneamente un único archivo (pantallas de texto y gráficos, impresora), por lo que se les nombra únicamente por el nombre de dispositivo.
Algunos dispositivos sólo pueden abrirse para entrada, para salida o para ambas.

	Entrada	Salida
cassette	X	X
p. de texto	-	X
p. de gráficos	-	X
impresora	-	X
cartucho	X	X
ud. de disco	X	X

Existen instrucciones que controlan los dispositivos sin utilizar archivos (CLOAD, CSAVE, LPRINT..)

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PRESET(20,20):PRINT #1,"que desea impri-
   mir ? "
40 A$=INPUT$(1)
50 PRINT #1,A$;
60 IF A$<> CHR$(13) GOTO 80
```


DJNZ Mnemónico de instrucción Z-80. Ejecuta un salto relativo si el registro B≠0, disminuyendo su valor en una unidad.
Se emplea en la creación de bucles en código máquina.

DRAW Instrucción para el trazado de rectas en las pantallas de gráficos. Actúa de acuerdo con las siguientes subinstrucciones:

- Sn — Factor de escala, n debe estar comprendido entre 0 y 255.
  Reducción, n entre 0 y 4.
  Escala real, n=4.
  Ampliación, n entre 4 y 255.
  Valor inicial S4.
- An — Giro del sistema de referencia, n debe estar comprendido entre 0 y 3. Gira el sistema de referencia en múltiplos de 90º.
  Valor inicial A0.
- Cn — Color, n debe estar comprendido entre 0 y 15. Especifica el código de color.
  Valor inicial A15 (blanco).
- Mx,y — Traza una recta desde la posición actual hasta la posición absoluta especificada por x,y.
  x comprendida entre 0 y 255, y comprendida entre 0 y 191.
  Es independiente de S.
- M±x,±y — Traza una recta desde la posición actual, tomada como origen de coordenadas hasta la posición relativa especificada por x,y.
  Tiene los mismos límites que Mx,y.
  Es dependiente de S.
- Un — Traza una recta en la dirección negativa del eje Y desde el punto actual

y con una longitud igual a n.
Es dependiente de S.
Si se omite, n toma el valor 1.

Dn Igual que Un pero en la dirección positiva de Y.

Rn Igual que Un pero en la dirección positiva de X.

Ln Igual que Un pero en la dirección negativa de X.

En Traza una recta desde la posición actual hasta otra de coordenadas relativas (n,-n).
Es dependiente de S.
Si se omite, toma el valor 1.

Fn Igual que En pero con coordenadas relativas (n,n).

Gn Igual que En pero con coordenadas relativas (-n,n).

Hn Igual que En pero con coordenadas relativas (-n,-n).

Todas las subinstrucciones, excepto S, A y C, desplazan la posición actual al último punto de la recta trazada. A todas estas subinstrucciones se les puede anteponer, optativamente, una de las dos subinstrucciones siguientes:

B Desplaza el punto actual, pero no traza la recta.

N Traza la recta, pero el punto actual no se desplaza.

```
DRAW  "BM127,96 NU40NR40"
```

Se puede expresar una subinstrucción o parte de ella mediante una variable alfanumérica, según los ejemplos siguientes:

```
A$="BM127,96U20R20D20L20"
DRAW A$
DRAW "S8XA$;"
```


Para sustituir n por una variable, se utiliza:
=variable;

En las subinstrucciones que tienen valores de inicialización: S, A y C, no son reinicializadas al utilizar CLEAR o RUN.

```
10 SCREEN 2
20 FOR I=1 TO 50 STEP 5
30 CLS
40 DRAW"BM35,115S=I;"
50 DRAW"U4F2E2D4"
60 DRAW"BM+2,0R3U2L3U2R3"
70 DRAW"BM+2,0F4Bm-4,0E4"
80 FOR J=0 TO 50:NEXT J
90 NEXT I
100 GOTO 10
```

El programa anterior traza las siglas MSX a distintas escalas.

eco de teclado Sonido semejante a un chasquido que se produce al pulsar las teclas.
Se puede desactivar mediante el tercer parámetro de SCREEN.

```
SCREEN ,,0        Activa el eco.
SCREEN ,,1        Desactiva el eco.
```

Para activar se emplea cualquier valor entre 1 y 255.

EI Un mnemónico de instrucción Z-80 que activa las interrupciones enmascarables.
Es contraria a DI.

ejecución automática de programas Los programas grabados en ASCII o en Código Binario se les puede hacer que se ejecuten automáticamente una vez que hayan sido cargados.
Existen las siguientes variantes:

```
LOAD "PROG1",R        Para cassette.
LOAD "nombre de dispositivo:PROG1",R
RUN "PROG1"           Para cassette.
RUN "nombre de dispositivo:PROG1"
BLOAD "nombre de dispositivo:PROG1",R
```

ejes Líneas rectas perpendiculares contenidas en un mismo plano, cuyo punto de corte es el origen de un sistema de coordenadas. (Véase coordenadas).

elemento Cada una de las variables individuales incluidas en una variable de conjunto.

```
DIM A(10,10)
```

Dimensiona un conjunto (tabla o matriz) de 11x11 elementos.

```
A(5,8)
```

Es el elemento perteneciente a la fila 6 y columna 9 de la matriz.

ELSE Es una parte opcional de la instrucción condicional IF ... THEN, que permite ejecutar una sentencia alternativa si la condición es falsa.

IF condición THEN sentencia 1 ELSE sentencia 2

Si (IF) la condición es verdadera, entonces (THEN) se ejecuta la sentencia 1.
Si es falsa (ELSE) se ejecuta la sentencia 2.

IF A=0 THEN PRINT "IGUAL A CERO" ELSE PRINT DISTINTO DE CERO"

Si A=0 se imprime IGUAL A CERO, si A≠0, se imprime DISTINTO DE CERO.
Al igual que IF ... THEN, cuando la sentencia sea GOTO número de línea, se puede omitir GOTO.

END (1) Da fin a la ejecución de un programa cerrando todos los archivos y sin visualizar mensaje de ruptura.
Se utiliza para separar subprogramas y subrutinas situadas al final de un programa, a fin de evitar la ejecución de dichas subrutinas una vez terminado el programa principal.
Para continuar la ejecución del programa, será necesario recurrir a una sentencia RUN o GOTO.

(2) Instrucción de ensamblador, da comienzo al proceso de ensamblaje.

ENDM Instrucción de ensamblador que se utiliza para finalizar la definición de un MACRO (Véase MAC).

ensamblador (1) Lenguaje de programación de bajo nivel. Cada una de las instrucciones de máquina del procesador se representa por un mnemónico, es decir, una clave que recuerda la acción de la instrucción.

De esta forma, se pueden escribir con cierta comodidad programas con instrucciones de máquina.

LD A,8

Es un mnemónico de una instrucción que asigna o "carga" en el registro A el valor 8.

(2) Un programa que traduce programas escritos en lenguaje ensamblador a código máquina. Suele incluir algunas opciones que facilitan la programación (etiquetas, macros ...).

ENT Instrucción de ensamblador que determina la dirección de ejecución del código máquina en función del valor de la expresión numérica que la acompaña.
En muchos casos, la dirección de comienzo de ejecución es diferente del comienzo del programa en código máquina.

entero Un número entero es el que carece de parte fraccionaria.
En BASIC MSX existe una precisión entera para representar este tipo de números, comprendidos entre -32768 y 32767, lo que precisa 2 bytes.

entrada de mando de juego El sistema MSX en su configuación mínima prevee una entrada para mando de juego (joystick), si bien muchos equipos presentan dos.
La misma entrada se utiliza para la conexión de raquetas (paddles), bola gráfica (trackball), tablero digitalizador (touchpad), ratón, etc. (Véase conector).

EOF Función BASIC que indica si se ha sobrepasado el marcador de fin de archivo.

Toma valores -1 en caso afirmativo, y 0 en caso contrario.
Se utiliza como condición para leer archivos de longitud desconocida. Utiliza como argumento el número de archivo.

```
IF EOF(1)=-1 THEN 100
```

Analiza si se ha llegado al fin de archivo, bifurcando a 100 en caso afirmativo.

EQU Instrucción de ensamblador. Va precedida de una etiqueta a la que asigna la dirección obtenida de la expresión numérica que le sigue.

```
WRTVDP: EQU#0047
```

Asigna a la etiqueta WRTVDP la dirección 47 en hexadecimal.

ERASE Instrucción para borrar las variables de matriz, esto nos permite utilizar el área de la memoria que ocupaban.
Utiliza como parámetro el nombre o inicial de variables de matriz.

```
10 DIM AX(100),BY(50,50)
.....
100 ERASE A,B
```

Elimina de memoria AX, BY y cualquier variable de matriz que empiece por A o B.

ERL Función que nos facilita el número de línea en la que se ha producido un error.
No utiliza argumentos.
Si no se produce error alguno, nos saldrá el valor cero, y si se produce en una sentencia en modo directo, aparecerá el valor 65335.
Es muy útil en depuración y comprobación de programas.

```
PRINT ERL
```


Imprime el número de línea de la sentencia que ha producido el error.

ERR Función que facilita el número de código de error que se ha producido. Aparecerá el valor cero cuando no se haya producido error.
Esta función no utiliza argumentos.

```
LITS
Syntax error
PRINT ERR
2
Ok.
```

El intento de ejecución de una instrucción mal escrita (LITS por LIST) provoca un error de sintaxis (cuyo mensaje de error es Syntax error) y con código 2.
(Véase mensaje de error).

ERROR Instrucción que produce el error especificado por su parámetro.

```
ERROR 2
Syntax error
```

Al ejecutar ERROR 2 aparece el mensaje de error correspondiente al código de error 2.
La instrucción ERROR también puede utilizarse para definir errores, es decir, situaciones que sean reconocidas como errores por el ordenador, a efectos, por ejemplo, del funcionamiento de ON ERROR.
Para ello, deben utilizarse códigos de error superiores a 59, pues los inferiores se emplean por BASIC.

```
10 ON ERROR GOTO 100
20 INPUT A
30 IF A=0 THEN ERROR 200
.....
90 END
```

```
100 IF ERR=200 THEN PRINT "Introduzca valores
    diferentes de cero"; RESUME 20
```

Si en la entrada de datos, se introduce un valor igual a cero, la instrucción ON ERROR bifurca la ejecución a la instrucción 100 para su tratamiento.

error de sintaxis Es un error frecuente en programación, que se produce cuando no se respetan las normas de unión de las distintas instrucciones funciones, operadores, etc. del lenguaje.
El MSX-BASIC no hace comprobación de sintaxis previa a la ejecución de los programas.
El mensaje de error correspondiente es "Syntax error".

especificación de archivo Es el nombre completo del archivo. Consta de tres partes:

- Dispositivo: El que soporta el archivo reconocido por su nombre clave que termine en dos puntos. Puede ser opcional.
- Nombre: Propiamente dicho, con un máximo de 6 caracteres, el primero alfabético
- Extensión: Para archivos en disco, va separado del nombre por su punto. Consta de tres letras. Si no se especifica, los archivos de programa en BASIC llevan la extensión .BAS

Cuando los archivos se llaman desde BASIC, van entre comillas. Cuando se llaman desde sistema, van sin comillas.

"DATOS" "PROG1" "CAS:DATOS" "CAS:PROG1"	Son archivos de programa o de datos (indistintamente) en MSX-BASIC de cassette.
A:PROG1.BAS	Archivo de programa en la unidad A, en BASIC de disco.
A:AUTOEXEC.BAT	Archivo autoejecutable al arranque del sistema que contiene a su vez instrucciones del DOS en la unidad A.
A:PROG1.BAT	Archivo modo BATCH que contiene a su vez instrucciones del DOS.

etiqueta Un símbolo o nombre empleado en ensamblador, al que se asigna generalmente la dirección de una instrucción, o una constante de hasta 16 bits.
Va seguida por dos puntos. El programa puede hacer referencia a esta dirección mediante la etiqueta:

```
FUNC: LD A,0
```

Asigna a la etiqueta FUNC la dirección donde comienza la instrucción:

```
LD A,0
```

EX Mnemónico de instrucción Z-80, que intercambia el contenido de un par de registros con el de otros.

```
EX DE,HL
```

Intercambia el contenido DE con el de HL.

EXP Función exponencial que ofrece el valor e^x. El valor máximo admisible para x es:
145.06286085862
El valor de e es: 2.71828182845888 siendo la base de los logaritmos neperianos.

```
PRINT EXP(1)
```

Imprime el valor de e en precisión doble:
2.718281 ...

exploración de teclado La exploración de teclado se realiza en el sistema MSX según la matriz:

COLUMNAS

<table>
<tr><th>FILAS</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>+</td><td>|</td><td></td><td>¥</td><td>-</td><td>=</td><td>9</td><td>8</td></tr>
<tr><td>2</td><td>B</td><td>A</td><td>-</td><td>/</td><td></td><td></td><td>|</td><td>*</td></tr>
<tr><td>3</td><td>J</td><td>I</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td></tr>
<tr><td>4</td><td>R</td><td>Q</td><td>P</td><td>O</td><td>N</td><td>M</td><td>L</td><td>K</td></tr>
<tr><td>5</td><td>Z</td><td>Y</td><td>X</td><td>W</td><td>V</td><td>U</td><td>T</td><td>S</td></tr>
<tr><td>6</td><td>F3</td><td>F2</td><td>F1</td><td>CODE</td><td>CAP</td><td>GRAPH</td><td>CTRL</td><td>SHIFT</td></tr>
<tr><td>7</td><td>RETURN</td><td>SELECT</td><td>BS</td><td>STOP</td><td>TAB</td><td>ESC</td><td>FS</td><td>F4</td></tr>
<tr><td>8</td><td>—</td><td>|</td><td>|</td><td>—</td><td>DEL</td><td>INS</td><td>HOME</td><td>SPACE</td></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

El valor del acumulador señala la fila a expresar y la rutina situada en &H141 carga en el mismo un byte cuyos ceros corresponden a la tecla pulsada de la matriz.

exponenciación Operación que consiste en elevar un número a un exponente. Si éste es entero equivale a multiplicar el número por sí mismo tantas veces como indique el exponente.

Se representa por el signo ^

2^3 = 2*2*2 = 8

exponente Si es entero, representa el número de veces que hay que multiplicar a un número por sí mismo.
En notación científica, los valores numéricos se expresan en función de potencias de 10, así:

2.78E2 = 2.78*10^2 = 278
2.78E-2 = 2.78*10^-2 = 2.78/100 = 0.0278

expresión Constante, variable o combinación de constantes, variables, operadores y funciones.
Puede ser numérica, alfanumérica o lógica.

96+90*SIN(N/4)	numérica
RIGHT$(A$,X)	alfanumérica
A<>0	lógica

EXX Semejante a EX, pero intercambiando conjuntamente BC con B'C', DE con D'E' y HL con H'L'.

falso Valor lógico. Se representa numéricamente por 0.

```
PRINT 3+(A<>0)
```

Imprime el valor 3 si A=0. Si A≠0 imprime 2.

FIELD Instrucción de BASIC de disco para el manejo de archivos de acceso directo.
Permite definir la longitud en caracteres de cada uno de los campos de un registro y asignar una variable para la transferencia de datos a cada campo.
El total de caracteres asignados no debe superar la longitud de registro.
Otras sentencias FIELD pueden asignar otra partición a un mismo registro utilizando variables diferentes.

```
FIELD #4, 30 AS A$,45 AS B$,4 AS S$
```

Reserva los 30 primeros caracteres del registro del archivo con número 4 para A$, los siguientes 45 para B$ y los últimos 4 para B$.

A$	B$	S$
1 30	31 75	76 79

Esta sentencia requiere una longitud de registro de al menos 79 bytes (30+45+4) o caracteres.
Las variables asignadas con FIELD no deben emplearse en otras sentencias de asignación.

figura móvil También conocida por sprite. Es una figura compuesta por un patrón de puntos dentro de una matriz de 8x8 ó 16x16 (Véase definición de sprite, SPRITE$).
Se le puede asignar posición, color y prioridad en caso de solapamiento. (Véase PUT SPRITE).

Las figuras móviles están activadas únicamente en los modos de pantalla 1, 2 y 3.
Existen dos tamaños para su visualización, además de los de definición (Véase SCREEN).
Los choques o solapamientos entre figuras móviles activan una de las interrupciones del sistema. (Véase ON SPRITE GOSUB, SPRITE ON/OFF/STOP).
Sin embargo, no puede conocerse qué sprites se han solapado, ni tampoco detectarse los choques sprites-fondo.
Alternativamente a la definición y control con SPRITE$ y PUT SPRITE se pueden modificar los valores de las tablas de la VRAM con VPOKE.
No pueden visualizarse más de cuatro figuras móviles en una misma línea horizontal. Si se hace, uno de los sprites desaparecerá.

FILES Instrucción de BASIC de disco. Visualiza los nombres de archivo contenidos en un disco.
Se pueden especificar nombres de archivo para comprobar si están en el disco.
Asímismo, pueden especificarse caracteres especiales para un tipo genérico de archivos.
El asterisco * sustituye cualquier conjunto de caracteres.
La interrogación ? sustituye a cualquier cáracter.

```
FILES "A:*.BAS"
```

Visualiza todos los archivos con extensión BAS de la unidad de disco A.

FIX Una función de BASIC que actúa sobre una expresión numérica truncándola. La función INT se diferencia de la función FIX en que actúa sobre los datos numéricos negativos devolviendo su parte entera:


```
FIX(11,6)       resulta  11
FIX(-11,6)      resulta -11
INT(-11,6)      resulta -12
```

La función FIX es equivalente a SGN(x)*INT(ABS(x))

FORMAT Instrucción de sistema operativo que da formato a un disco.
Mediante esta acción se comprueba el disco, evitando las zonas deterioradas, se reserva espacio para el directorio y se prepara para su uso.
Debe hacerse previamente a la utilización de discos nuevos. Si se hace con discos usados, se borrará toda la información contenida en ellos.
Desde sistema, hacemos:

```
FORMAT A:
```

y desde BASIC:

```
CALL FORMAT A:
```

FOR...NEXT Un conjunto de instrucciones de BASIC:

```
FOR variable = valor inicial TO
valor final STEP incremento
....................
instrucciones interiores del bucle
....................
NEXT variable
```

Permiten la creación de bucles, es decir, la repetición de las sentencias contenidas entre FOR y NEXT con la variable tomando valores desde el inicial hasta el final con aumentos (o disminuciones) del valor de incremento (puede ser positivo o negativo, en este último caso, el valor inicial debe ser mayor que el final).
El valor de la variable debe estar comprendido por lo tanto, entre el valor inicial y el final.

El contador del bucle se encuentra en NEXT, donde se incrementa (o decrementa) el valor de la variable. La condición se comprueba en FOR, si se cumple, se repiten las sentencias interiores, si no, continua la ejecución del programa a partir de la sentencia que sigue a NEXT.
La instrucción STEP puede omitirse, en cuyo caso el incremento es +1. Si se toma STEP 0, el bucle se repite indefinidamente.

```
10 CLS
20 FOR A=1 TO 10
30 PRINT A
40 NEXT A
```

Presenta en pantalla los diez valores 1 ... 10 asignados en cada pasada del bucle a la variable A. Se puede prescindir de la variable en NEXT, si bien en el caso de disponer varios bucles, unos dentro de otros (véase anidados), debe haber tantos NEXT como FOR.

```
correcto              incorrecto
FOR A=1 TO 10         10 FOR A=1 TO 10 STEP -1
20 PRINT A            20 PRINT A
30 NEXT A             30 NEXT A
```

El programa incorrecto se debe al incremento negativo con límites crecientes.

```
correcto                    incorrecto
10 FOR A=1 TO 1 STEP -1     10 FOR A=10 TO 1
20 PRINT A                  20 PRINT A
30 NEXT A                   30 NEXT A
```

El programa incorrecto se debe al incremento positivo con límites decrecientes.

```
10 FOR A=32 TO 255
20 PRINT CHR$(A);" ";
30 NEXT A
```

Imprime el juego de caracteres del ordenador.

FRE Una función BASIC que ofrece el número de bytes libres, tanto en el área de programación BASIC, como en la de almacenamiento de cadenas.
En ordenadores de 64K de RAM:

```
PRINT FRE(0)      imprime 28815
PRINT FRE("")     imprime 200 (valor por
                  omisión), y como máximo
                  255, que corresponde al
                  área de almacenamiento de
                  cadenas máximo.
```

Se utiliza para conocer la cantidad de memoria que nos queda libre para programas y datos.
A diferencia de otros ordenadores, los sistemas MSX no consumen memoria de usuario para la pantalla. Esta cuenta con una memoria separada controlada directamente por el VDP.

función Una transformación de una valor dado (argumento) numérico, alfanumérico o lógico en otro de distinta o similar naturaleza según una relación previamente establecida.
Algunas funciones no precisan argumentos al estar relacionadas con valores internos del ordenador, producidos por teclado, etc.
Suelen distinguirse los siguientes tipos:

1. Numéricas: retornan valores numéricos. Pueden ser:
   1.1. Matemáticas: operan con números.
   1.2. Alfanuméricas: operan con cadenas.
   1.3. Entrada/Salida.
   1.4. Otras.
2. Alfanuméricas: retornan valores alfanuméricos. Pueden ser:
   2.1. Numéricas: operan con números.
   2.2. Alfanuméricas: operan con cadenas.
   2.3. Entrada/Salida.
   2.4. Varias.

3. Definidas por el usuario.
   (Véase DEF FN).

funciones trigonométricas Se dice de las funciones que tienen su origen en transformaciones trigonométricas.
Son en su mayoría funciones numéricas matemáticas.
El MSX-BASIC reconoce las siguientes:

ATN(X)	arcotangente de X
COS(X)	coseno de X
SIN(X)	seno de X
TAN(X)	tangente de X

Operan con argumentos en radianes (excepto ATN(X) que devuelve el ángulo en radianes).
Mediante ATN podemos definir el valor de PI:

```
PI=4*ATN(1)
```


ganchos Algunas rutinas de la ROM realizan llamadas (instrucción CALL) a determinadas direcciones de la RAM (&HFD9A-&HFFC9) que en la inicialización del sistema contienen la instrucción RET, por lo que retornan a la ROM.
Estas direcciones de la RAM pueden modificarse con instrucciones diferentes de RET, por ejemplo, una llamada (CALL) o un salto JP incondicional a otra zona de memoria.
Así se pueden "enganchar" rutinas propias o modificar la ejecución prefijada de los existentes.
Cada gancho está compuesto de cinco instrucciones RET, por lo que se pueden insertar instrucciones complejas (CALL o SP requieren 3 bytes).
Al pulsar cualquier tecla, se produce una llamada a la posición &HFDD1, si en esta posición y siguientes situamos una llamada a una rutina propia, esta rutina se ejecutará cada vez que pulsamos una tecla.

generador programable de sonido Es el circuito AY-3-8910 de General Instruments, un clásico dentro de su gama.
Puede generar tono o ruido en tres canales y con controles de volumen, frecuencia (fino y grueso) y envolvente.
Para ello se utilizan 16 registros, según la siguiente relación:

Ajuste fino frecuencia tono:

R 0	canal A
R 2	canal B
R 4	canal C

Ajuste grueso frecuencia tono:

R 1	canal A
R 3	canal B
R 5	canal C

Frecuencia de ruido:
R 6

Control de conexión/desconexión:
R 7

Amplitud y envolvente:
R 8 canal A
R 9 canal B
R 10 canal C

Ajuste fino frecuencia envolvente:
R 11

Ajuste grueso frecuencia envolvente:
R 12

Forma envolvente:
R 13

Puerto A:
R 14

Puerto B:
R 15

Estos registros se asignan con la instrucción BASIC SOUND, siendo de aplicación lo allí expuesto para el control de los registros.

gestión de memoria El procesador Z-80 puede direccionar simultáneamente 64K de memoria.
El intérprete BASIC y el Sistema Operativo ocupan los primeros 32K (direcciones &H0000 a &H80000) y reservan como área de trabajo. La zona final comprendida entre &HF380 a &HFFFF.
La especificación mínima fija 8K de RAM de usuario si bien en la práctica se suele disponer de 32K o 64K (en este caso 32K son superpuestos por la RAM).

Para un sistema de 32 ó 64K, la RAM libre es de 28K (concretamente 28815 bytes).
Los 64K totales se pueden dividir en 4 páginas de 16K. Cada página va numerada 0, 1, 2 y 3.
Pueden existir cuatro grupos de cuatro páginas denominados ranuras de forma que cada página de las cuatro que forman 64K totales, puedan proceder de una de las ranuras.
De esta forma, aunque el procesador sólo direcciona 64K, estos pueden proceder en grupos de 16K de un total de 256K. Una página solamente podrá aparecer ante el procesador en la misma posición que tenía en la ranura. A su vez, cada ranura puede controlar otras cuatro, por lo que la memoria total podría elevarse a 1 Mbyte o Megabyte.

GET Instrucción de BASIC de disco, utilizada en la lectura de registros de archivos aleatorios

```
GET #4,27
```

Lee el registro 27 del archivo abierto con el número de canal 4, asignando a las variables especiales definidas por FIELD, el contenido del registro, de acuerdo con la partición expresada en la misma.

grado Unidad angular. Pueden ser sexagesimales, los más utilizados, que equivales a PI/180 radianes; o bien, centesimales, que equivalen a PI/200 radianes.
Todas las funciones trigonométricas del ordenador manejan radianes, por lo que se hace precisa su conversión, cuando deben emplearse grados.

GOSUB...RETURN Es un juego de instrucciones BASIC.

```
GOSUB número de línea
```

Guía el flujo del programa hacia una subrutina situada en una línea especificada.

Mediante la sentencia RETURN colocada al final de la subrutina, se devuelve el flujo del programa a la línea inmediatamente posterior al GOSUB, o a un número de línea especificada en RETURN. Si se ejecuta RETURN sin haber pasado anteriormente por un GOSUB, se produce un error.

    RETURN without GOSUB

Se puede hacer una llamada a una subrutina desde otra, y a su vez desde otra, y así sucesivamente, dependiendo de la capacidad de la pila del ordenador (Véase pila).
Generalmente, para separar cada una de las subrutinas del programa principal, se utiliza la instrucción END (Véase END).
La utilización de subrutinas en BASIC permite simular programación estructurada.
La utilización de GOSUB aisladamente, produce una bifurcación incondicional a la subrutina. Si la utilizamos en unión de IF en la forma:

    IF condición THEN GOSUB nº línea ELSE GOSUB nº línea

Se produce una bifurcación condicional a la subrutina.

GOTO Una instrucción BASIC que efectúa una bifurcación incondicional en el programa hacia el número de línea indicado. La utilización de esta instrucción, en modo directo, ejecuta el programa a partir de la línea especificada a continuación de GOTO, pero sin inicializar las variables, como haría RUN número de línea.

```
100 GOTO 100
```

Produce un bucle indefinido al repetir sucesivamente la instrucción 100. Aunque aparentemente no tenga sentido, esta sentencia es muy útil para mantener en pantalla los dibujos creados en los modos de gráficos.

Si no se utilizara, se pasaría al modo de texto, borrándose los dibujos.
Las bifurcaciones pueden hacerse condicionales mediante:

    IF condición THEN GOTO nº línea ELSE nº línea

En este caso, puede prescindirse de GOTO, así:

    IF condición THEN nº línea ELSE nº línea

gráficos Para el trazado de gráficos existen dos modos de pantalla especialmente diseñados, el modo de gráficos de alta resolución y el modo multicolor.
También se dispone de numerosas instrucciones de gráficos: PSET, PRESET, LINE, CIRCLE ... , e incluso de un sublenguaje completo para gráficos con instrucciones propias, factores de escala, giro, etc. (DRAW).
También se pueden generar gráficos de una forma restringida en los modos de texto, mediante la utilización de los caracteres gráficos, situados con la instrucción LOCATE.

```
10 SCREEN 2
20 COLOR 4,7,15:CLS
30 LINE(0,150)-(255,191),4,BF
40 CIRCLE(200,55),50,11,,,1.2
50 PAINT(200,55),11
60 GOTO 60
```

Dibuja in cielo azul claro, un mar azul oscuro y un sol amarillo.

gráficos alta resolución Se pueden generar gráficos de alta resolución mediante el modo de pantalla obtenido con SCREEN 2. Este modo dispone de 256x192 puntos direccionables independientemente, aunque con una pequeña limitación en su color que consiste en que cada segmento horizontal de 8 puntos puede contener únicamente dos colores diferentes.

HALT Mnemónico de instrucción Z-80 que detiene la ejecución de programa por el procesador hasta la generación de una interrupción.

HEX$ Una función alfanumérica de BASIC que entrega una cadena con el equivalente en base hexadecimal de expresiones numéricas, en cualquier base, con su prefijo correspondiente:

```
PRINT HEX$(&B11111111)        resulta FF
PRINT HEX$(&O15523)           resulta 1B53
PRINT HEX$(255)               resulta FF
```

Véase notación hexadecimal.

IF...THEN...ELSE Instrucción que permite la ejecución de diferentes sentencias en función del cumplimiento o no de las condiciones determinadas por el usuario.
En caso de ser la condición verdadera, se ejecutarán las sentencias precedidas por THEN; si son varias, se separarán por dos puntos.
Si el resultado de la condición es falso, bifurcará a las sentencias posteriores a ELSE, y en caso de omisión de éste, a la línea siguiente.
Si se va a realizar una bifurcación del flujo del programa, no es necesario utilizar GOTO después de THEN y ELSE. También puede sustituirse THEN por GOTO.
Pueden anidarse varios IF ... THEN:

```
IF ... THEN ... IF ... THEN ... ELSE
```

En este caso, si la primera condición es falsa, el flujo transferirá a la línea siguiente.

```
100 IF A=0 THEN PRINT "A es igual a cero"
    ELSE PRINT "A es distinto de cero"
```

Esta sentencia IF comprueba si A=0 imprimiendo el mensaje correspondiente en cada caso.
Una sentencia semejante puede bifurcar hacia uno u otro número de línea la ejecución del programa.

```
100 IF A=0 THEN GOTO 150 ELSE GOTO 200
```

O bien, prescindiendo de GOTO:

```
100 IF A=0 THEN 150 ELSE 200
```

Bifurcamos a 150 si A=0 ó a 200 si $A \neq 0$.
La condición puede ser compuesta, es decir, puede constar de más de una condición simple unidas mediante operadores lógicos, cuyo resultado depende de las tablas de verdad de dichos operadores.

```
IF A=0 AND B=2 THEN PRINT "A es cero y B es
dos"
```

En este caso, la condición es verdadera si A=0 y B=2, simultáneamente.

```
10 INPUT A,B,C
20 IF A=1 THEN IF B=2 AND C=A+B THEN PRINT
   "CORRECTO";: GOTO 40 ELSE 50
30 PRINT "FALTA":GOTO 10
40 PRINT "PRIMER PREMIO":END
50 PRINT "SEGUNDO PREMIO":GOTO 10
```

El programa anterior solicita tres números que deben ser: uno, dos y tres. Si la sentencia es correcta, se produce el primer premio, si es parcialmente correcta el segundo, y falta en caso contrario.

IM Mnemónico de instrucción Z-80 que fija el modo de interrupción del procesador.
Existen tres modos de interrupción que permiten a un dispositivo externo forzar la ejecución de determinadas subrutinas.

- IM0 Permite realizar instrucciones RST o CALL.
- IM1 Realiza la inicialización del sistema a partir de la instrucción &H38.
- IM2 Permite acceder hasta 128 rutinas diferentes.

Todas ellas terminan con una instrucción RETI.

impresora Dispositivo que permite la impresión de listados de programas, datos y resultados.
Existen varias instrucciones para su utilización directa (LPRINT, LLIST ...), si bien también puede ser utilizada como un dispositivo más con el nombre de LPT:
El sistema emplea el protocolo Paralelo-Centronics
Se admiten impresoras Centronics standard, o bien, específicas del sistema MSX, lo que se determina con el quinto parámetro de SCREEN.

```
SCREEN,,,,0
```


Dispone el ordenador para impresoras MSX, cualquier valor distinto de cero se emplea para impresoras standard. La diferencia estriba en que las impresoras del sistema MSX contienen la definición de caracteres gráficos iguales a los de pantalla. En las restantes, dichos caracteres no coinciden, por lo que si seleccionamos esta opción, el ordenador transmite espacios en lugar de caracteres gráficos.

IN Mnemónico de instrucción Z-80 que lee uno de los puertos de E/S y carga el valor obtenido en alguno de los registros.
(Véase puertos de E/S).

INC Mnemónico de instrucción que aumenta en una unidad alguno de los registros, par de registros o contenido de una posición de memoria.
Si el registro contiene el valor 0 la bandera Z adquiere el valor 1, y 0 en caso contrario.

IND Mnemónico de instrucción Z-80 que lee el puerto de E/S cuyo número se encuentra en el registro C, carga el valor en la posición de memoria HL disminuyendo su valor y el de B en una unidad.

INDR Semejante a IND pero con repetición hasta que B se hace cero.

INI Semejante a IND pero el par HL se incrementa en vez de disminuir.

inicialización (1) Proceso de asignación inicial de variables. Suele realizarse al principio de un programa, señalando los valores de partida y declarando tipos y precisiones.

(2) Procedimiento seguido por el ordenador para su puesta en marcha o reposición.
Se realiza mediante una llamada a la dirección &HOO.

INIR Semejante a INDR pero el par HL se incrementa en vez de disminuir.

INKEY$ Función que dá el primer cáracter que se encuentre en la memoria intermedia del teclado, si está vacía, resulta la cadena nula.

```
10 A$=""
20 A$=INKEY$
30 IF A$="" THEN 20
40 PRINT A$;
50 GOTO 20
```

Este programa actúa como una máquina de escribir.

INP Una función de BASIC que lee uno de los puertos de E/S. Utiliza como argumento la dirección del puerto (Véase puerto de E/S).
Retorna un valor entre 0 y 255.

INPUT Instrucción que detiene el flujo del programa en espera de introducción de datos a través del teclado, y asigna estos a las variables especificadas.
En pantalla se visualizará el carácter "?"
Se puede incluir un comentario, éste ha de ir entre comillas y terminado por punto y coma.

```
INPUT "comentario";A
```

Pueden asignarse más de una variable, en cuyo caso se separarán por comas en la sentencia y la entrada de datos podrá realizarse de uno en uno pulsando RETURN después de cada uno de ellos, o todos a la vez, separándolos con comas.

Si la variable no ha sido asignada con anterioridad y el usuario no introduce datos antes de pulsar RETURN, se asignará el valor 0 a las variables numéricas y la cadena nula a las alfanuméricas.
En el caso de que éstas sí hubiesen sido asignadas, estas variables quedarán inalteradas.

```
10 INPUT "introduzca A,B,C$";A,B,C$
20 PRINT A;B;C$
30 GOTO 10
```

INPUT$ Función que asigna a una variable el número especificado de caracteres, introducidos desde el teclado.

```
10 A$=INPUT $(4)
20 PRINT A$
```

Mientras se introducen los caracteres, en pantalla no se visualizará nada. Tampoco se visualiza el caracter "?" de la sentencia INPUT.
Mediante esta función no podrán asignarse cadenas de más de 255 caracteres de longitud.
Puede leerse el número especificado de caracteres de un archivo mediante la variante:

```
A$=INPUT$(25,#1)
```

Leerá 25 caracteres del archivo 1. Este archivo ha de abrirse previamente con una instrucción OPEN

INPUT# Instrucción que asigna a una variable los datos leidos en un archivo. El archivo ha de estar abierto mediante una sentencia OPEN.
Los datos serán asignados a las variables según el formato que tengan mediante los separadores.
Así, los datos que en el archivo se encuentran separados por comas, por el código de retorno o por el código de avance de línea, serán separados al ser leidos por INPUT#.
Los datos numéricos podrán haber sido separados

también por el cáracter espacio. Ejemplo:

```
10 PRINT"1-grabar":PRINT"2-leer"
20 INPUT Z1
30 ON Z1 GOTO 40,110
40 OPEN "cas:DAT1" FOR OUTPUT AS #1
50 FOR I=1 TO 5
60 READ A$
70 PRINT#1,A$;",";
80 NEXT I
90 CLOSE #1
95 END
100 DATA cassette,cartucho,disco,impresora,
ordenador MSX
110 OPEN "cas:DAT1"FOR INPUT AS #1
120 FOR I=1 TO 5
130 INPUT #1,A$(I)
140 PRINTA$(I)
150 NEXT I
160 CLOSE #1
```

Los datos alfanuméricos, si en el archivo se encuentran separados entre comillas, serán tenidos en cuenta como un único dato. Una forma de que ocurra esto es escribiendo en el archivo mediante:

```
PRINT#1,CHR$(34)+"amigo,pepe"+CHR$(34)
```

De esta forma la coma no actuará como separador.
Para este caso también se puede utilizar:

```
LINE INPUT#
```

INSTR Función que tras haber localizado una cadena especificada en otra, nos dá la posición de comienzo de su ubicación. La dirección de conteo es de izquierda a derecha.

```
INSTR (A$,B$)
```

donde B$ es la cadena a buscar en A$.
También está permitido darle la posición inicial de búsqueda


```
PRINT INSTR(6,"hola y adios","a")
```

dará 8, que es la posición de la segunda "a" de la cadena.
Si la cadena a buscar no está, o el número de inicio de búsqueda es mayor que la longitud de la cadena, resultará el valor 0.

```
10 PRINT "DEME UNA FRASE";A$
20 PRINT"CARACTER A BUSCAR ";:B$=INPUT$(1):
PRINT B$:PRINT
30 C=0
40 C=INSTR(C+1,A$,B$)
50 IF C>LEN(A$) OR C=0 THEN END
60 PRINTB$;" EN LA POSICION";C
70 GOTO 40
```

instrucción Es una palabra clave que realiza una acción determinada: mostrar un caracter en pantalla, dibujar un gráfico, grabar un sonido, etc.
Es la unidad básica para el desarrollo del programas.
Una instrucción completa con todos sus parámetros es una sentencia.

INT Función que calcula la parte entera de un dato numérico. Es igual a la función FIX, salvo en los datos numéricos negativos, en los cuales no coinciden. Por ejemplo:

```
INT(-3,2)        resulta -4
FIX(-3,2)        resulta -3
```

Mientras INT toma el valor entero inmediatamente inferior, FIX toma el superior.
La equivalencia entre ambas funciones viene dada por la siguiente expresión:

```
FIX(X)=SGN(X)*INT(ABS(X))
```

interfaz Es un circuito que permite la comunicación del procesador con el mundo exterior.
Suele disponer de algún tipo de conector exterior.

interfaz de cassette Actúa con modulación en frecuencia (modo FSK) controlada por programa.
Dispone de dos velocidades de transmisión/recepción:

1200 baudios
2400 baudios

que se seleccionan en transmisión por SCREEN en su cuarto parámetro.
En recepción el ajuste se produce automáticamente.
El conector es del tipo DIN de 8 patillas, controlando incluso el funcionamiento del motor.

interfaz programable para periféricos
Utiliza el circuito 8255 de Intel, que realiza la exploración de teclado, gestión de memoria, cassette y piloto de mayúsculas, entre otras funciones.

intérprete Es un tipo de lenguaje de programación que se caracteriza por traducir cada instrucción al lenguaje de máquina en el momento de su ejecución.
Siempre que el programa se ejecuta será traducido, en consecuencia, es más lento que los lenguajes compilados, aunque tiene la ventaja de que para comprobar un programa no es necesario proceder a su compilación.

interrupción Proceso mediante el cual se obtiene una bifurcación del programa principal debido a una acción externa o generada por el propio ordenador.
Esta interrupción está definida por las siguientes instrucciones:

ON KEY GOSUB
ON STRIG GOSUB
ON STOP GOSUB
ON SPRITE GOSUB
ON INTERVAL=n GOSUB

las cuales están condicionadas por:

KEY( )ON/OFF/STOP
STRIG( )ON/OFF/STOP
STOP ON/OFF/STOP
SPRITE ON/OFF/STOP
INTERVAL ON/OFF/STOP

En donde ON dá validez a la interrupción, OFF la cancela, y STOP la suspende y memoriza hasta encontrarse con un ON o RETURN que la ejecuta.

INTERVAL ON Instrucción que dá validez al proceso de interrupción declarado previamente por:

ON INTERVAL=n GOSUB

(Véase interrupción).

INTERVAL OFF Instrucción que cancela el proceso de interrupción declarado previamente por:

ON INTERVAL=n GOSUB

(Véase interrupción).

INTERVAL STOP Instrucción que suspende y memoriza el proceso de interrupción declarado previamente por:

ON INTERVAL=n GOSUB

hasta encontrarse con un ON o RETURN que ejecute el proceso de interrupción.
(Véase interrupción).

joystick Es un mando de juego que consta de una palanca móvil, con ocho posiciones y dos disparadores.
El sistema admite hasta dos entradas para estos mandos además de las teclas de cursor y barra de espacio que permiten simular la acción de un tercero.
Las instrucciones asociadas a su manejo son: STRIG y STICK.
Existe un dispositivo semejante, la raqueta o paddle, controlado mediante PDL.

JP Un mnemónico de instrucción Z-80 que realiza un salto a una dirección determinada a partir de la cual prosigue la ejecución del programa.
La dirección puede especificarse directamente:

JP 60000	saltará a la posición 60000.
JP(60000)	saltará a la posición almacenada en 60000 y 60001.

Existen saltos condicionales:

JP Z	salta si el último resultado es cero.
JP NZ	salta si el último resultado es distinto de cero.
JP C	salta si la bandera de exceso es 1.
JP NC	salta si la bandera de exceso es 0.

JR Un mnemónico de instrución Z-80 que realiza un salto relativo sumando o restando el valor que le sigue a la posición actual. Dicho valor es un único byte, que es considerado en notación de complemento de dos, por lo que se considera negativo si su bit 7 es 1, y positivo si es cero.

Un valor negativo realiza un salto hacia atrás, y uno positivo hacia delante.

```
JR -2
```

Debe evitarse, porque salta a la misma instrucción lo que crearía un bucle indefinido.
Existen instrucciones JR condicionales:

JR Z	Salta si el último resultado es cero.
JR NZ	Salta si el último resultado es distinto de cero.
JR C	Salta si la bandera de exceso es 1.
JR NC	Salta si la bandera de exceso es cero.

juego de caracteres Es el conjunto de cacarcteres que pueden visualizarse por el ordenador. Pueden obtenerse mediante el programa:

```
10 FOR N=32 TO 255
20 PRINT CHR$(N);" ";
30 NEXT N
40 FOR N=1 TO 31
50 PRINT CHR$(1)+CHR$(N);" ";
60 NEXT N
```

A continuación se relacionan junto con su código ASCII en decimal:

0	(nula)	9	○	18	┬
1	☺	10	◙	19	┤
2	☻	11	♂	20	├
3	♥	12	♀	21	┼
4	♦	13	♪	22	│
5	♣	14	♫	23	─
6	♠	15	☼	24	┌
7	•	16	┼	25	┐
8	◘	17	┴	26	└

27	┘	63	?	99	c
28	X	64	@	100	d
29	╱	54	A	101	e
30	╲	66	B	102	f
31	+	67	C	103	g
32	(espacio)	68	D	104	h
33	!	69	E	105	i
34	"	70	F	106	j
35	#	71	G	107	k
36	$	72	H	108	l
37	%	73	I	109	m
38	&	74	J	110	n
39	'	75	K	111	o
40	(	76	L	112	p
41	)	77	M	113	q
42	*	78	N	114	r
43	+	79	O	115	s
44	,	80	P	116	t
45	-	81	Q	117	u
46	.	82	R	118	v
47	/	83	S	119	w
48	Ø	84	T	120	x
49	1	85	U	121	y
50	2	86	V	122	z
51	3	87	W	123	{
52	4	88	X	124	¦
53	5	89	Y	125	}
54	6	90	Z	126	~
55	7	91	[	127	
56	8	92	\	128	ç
57	9	93	]	129	ü
58	:	94	^	130	é
59	;	95	_	131	â
60	<	96	`	132	ä
61	=	97	a	133	à
62	>	98	b	134	a

135	ç	171	½	207	▶
136	ê	172	¼	208	◀
137	ë	173	¡	209	⧗
138	è	174	«	210	⧓
139	ï	175	»	211	▌
140	î	176	Ã	212	▐
141	ì	177	ã	213	▪
142	Ä	178	Ĩ	214	▖
143	Å	179	ĩ	215	▓
144	É	180	Õ	216	Δ
145	æ	181	õ	217	‡
146	Æ	182	Ũ	218	ω
147	ô	183	ũ	219	█
148	ö	184	Ĳ	220	■
149	ò	185	ĳ	221	▌
150	û	186	¾	222	▐
151	ù	187	~	223	■
152	ÿ	188	◇	224	α
153	Ö	189	‰	225	β
154	Ü	190	¶	226	Γ
155	¢	191	§	227	π
156	£	192	▀	228	Σ
157	¥	193	▚	229	σ
158	Pt	194	█	230	μ
159	ƒ	195	▬	231	γ
160	á	196	▬	232	ɸ
161	í	197	█	233	θ
162	ó	198	▌	234	Ω
163	ú	199	▞	235	δ
164	ñ	200	▐	236	∞
165	Ñ	201	▜	237	ø
166	ª	202	▐	238	∈
167	º	203	⁄⁄	239	∩
168	¿	204	\\	240	≡
169	⌐	205	▼	241	±
170	¬	206	▲	242	≥

243	≤	248	°	252	ⁿ
244	⌠	249	•	253	²
245	⌡	250	–	254	■
246	÷	251	√	255	cursor
247	≈				

KEY Instrucción BASIC que asigna la cadena deseada a la tecla de función especificada mediante su número. La cadena puede tener hasta 15 caracteres como máximo, aunque en pantalla sólo se visualizarán 6.
La cadena se mantendrá asignada mientras no hagamos una reposición del sistema o desconectemos de la red, en cuyo caso se inicializan todas las teclas de función con los valores habituales.
Si hacemos:

```
KEY 1,"RUN 200"+CHR$(13)
```

Con sólo pulsar F-1 se ejecutará el programa a partir de 200.

KEY LIST Instrucción BASIC que nos lista ordenadamente, en pantalla, el contenido de todas las teclas de función.

KEY n ON Instrucción que dá validez al proceso de interrupción declarado previamente por:
ON KEY GOSUB
(Véase interrupción).

KEY n OFF Instrucción que cancela el proceso de interrupción declarado previamente por:
ON KEY GOSUB
(Véase interrupción).

KEY n STOP Instrucción que suspende y memoriza el proceso de interrupción declarado previamente por:
ON KEY GOSUB
hasta encontrarse con un ON o RETURN que ejecuta el proceso de interrupción.
(Véase interrupción).

KEY ON Instrucción BASIC que visualiza en pantalla el contenido de las teclas de función en la línea 24 de la pantalla.
Solamente se muestran seis caracteres de los valores asignados a las teclas F1-F5.
Si pulsamos SHIFT se visualizan los valores de F6-F10.

KEY OFF Instrucción BASIC que borra de la pantalla el contenido de las teclas de función.
Permite imprimir en la línea 24 de pantalla.

KILL Es una instrucción BASIC de disco, que elimina el archivo especificado a continuación de la instrucción:

    KILL "A:PROG1.BAS"

Borra el programa BASIC denominado PROG1 en la unidad A.
Si no se especifica unidad, se toma la asumida por omisión. Es semejante a DELETE (instrucción de sistema).

lápiz de luz Dispositivo en forma de lápiz que permite determinar posiciones de pantalla, siendo especialmente adecuado para el diseño gráfico.

LD Un mnemónico de instrucción Z-80 que carga un valor en un registro o en memoria.
Presenta diversas formas:

LD A,28	Carga el valor 28 en A.
LD A,(60000)	Carga en A el valor contenido en la dirección de memoria 60000.
LD HL,(60000)	Carga en HL el valor contenido en 60000 y 60001.
LD(60000),A	Carga en 60000 el valor contenido en A.
LD L,A	Carga en L el valor de A.

LDD Un mnemónico de instrucción Z-80 que carga el valor contenido en la posición señalada por HL en la posición determinada por DE.
A continuación HL, BC y DE disminuyen en una unidad.

LDDR Un mnemónico de instrucción Z-80 semejante a LDD pero con repetición hasta que BC sea cero. Realiza, por lo tanto, una transferencia de bloques.

LDI Un mnemónico de instrucción Z-80 semejante a LDD pero aumentando HL y DE en una unidad y disminuyendo BC.

LDIR Un mnemónico de instrucción Z-80 semejante a LDI pero con repetición hasta que BC sea cero. Realiza, al igual que LDDR una transferencia de bloques.

LEFT$ Función alfanumérica de BASIC que dá el número especificado de caracteres, contando de izquierda a derecha, de una cadena alfanumérica.

    PRINT LEFT$("ORDENADOR MSX",6)

Obtiene ORDENA.
También es válido:

    LEFT$(A$,n)   Para n comprendido entre 0 y 255.

Un carácter gráfico ocupa dos posiciones. Si n=0 se obtiene la cadena nula. Si n es mayor que la longitud de A$ se obtiene A$ y no se añaden espacios. Si n es decimal se omitirán los números siguientes al punto decimal.

LEN Función numérica de BASIC que dá la longitud de una cadena.

    PRINT LEN("ORDENADOR MSX")

Obtiene el valor 13.
Los caracteres gráficos se contabilizarán como dos caracteres.

lenguaje ensamblador Las instrucciones del procesador se representan mediante abreviaturas que recuerdan las acciones que desarrollan. Estas abreviaturas, fáciles de recordar, se denominan mnemónicos y constituyen el lenguaje ensamblador. Para su carga en el ordenador se precisa de un programa especial denominado ensamblador.

lenguaje máquina Es el lenguaje "entendido" por el procesador del ordenador. En último extremo, está formado por una secuencia de impulsos eléctricos que pueden representarse para su programación mediante series de unos y ceros.
Debido a la longitud de estas series y la posible aparición de errores, se sustituyen por una

representación hexadecimal.
El procesador Z-80 utiliza este lenguaje (que es distinto en cada tipo de procesador). Su versión consta de 158 tipos de instrucciones agrupadas en los siguientes grupos:

- Operaciones de carga.
- Operaciones aritméticas.
- Operaciones lógicas.
- Operaciones de rotación y desplazamiento.
- Operaciones de bits.
- Operaciones de bifurcación y rutinas.
- Operaciones de transferencia de bloques.
- Operaciones de control.
- Operaciones Entrada/Salida.

LET Instrucción BASIC que se usa para asignar el valor de una determinada expresión a una variable. Esta instrucción se puede omitir en el BASIC MSX, simplemente escribiendo:

    variable = expresión

Ejemplo:

```
10 FOR I=1 TO 5
20 LET X=I^2
30 PRINT X
40 NEXT I
```

Imprime los valores 1, 4, 9, 16, 25.

LINE Instrucción BASIC para el trazado de rectas y rectángulos en las pantallas de gráficos.
Así:

    LINE(30,30)-(100,100),1

Trazará una recta en color negro entre los puntos de coordenadas especificadas. El color puede omitirse, en cuyo caso tomará el color del primer plano.

    LINE(30,30)-(100,100),,B

Trazará un rectángulo cuya diagonal está definida por los puntos de coordenadas especificadas.

```
LINE(30,30)-(100,100),,BF
```

Coloreará el rectángulo.
También puede utilizarse en la forma:

```
LINE(30,30)-(100,100)
LINE-(200,50)
```

En donde el segundo LINE trazará una recta desde el último punto anterior, en este caso (100,100) hasta el punto (200,50). Admite todas las variantes anteriores, color, rectángulo y coloreado.
En todas las variantes anteriores se puede hacer uso de STEP para coordenadas realtivas, en la forma:

```
STEP(X,Y)
```

(Véase STEP).

LINE INPUT Instrucción BASIC semejante a INPUT pero con las siguientes características:

- Permite introducir comas en las cadenas.
- No aparece el caracter ?
- Sólo sirve para la asignación de una sola variable.

LINE INPUT # Instrucción que asigna a una variable alfanumérica una cadena de hasta 254 caracteres de un archivo.
En este caso, las comas no actuarán como separador.

```
110 OPEN "cas:DAT1" FOR INPUT AS #1
120 LINE INPUT #1,B$
130 PRINTB$
140 CLOSE #1
```

linea Una línea en BASIC consta de un número de línea comprendido entre 0 y 65529 y una sentencia, es decir, una instrucción completa con sus parámetros.


```
10 PRINT "ORDENADOR MSX"
```

linea actual Se denomina línea actual a la línea en edición o ejecución.
Si se produce un error en el programa, LIST listará la línea actual, en ejecución, en el momento de producirse el error.

LIST Instrucción BASIC, para utilizar en modo directo, que visualiza la línea o líneas especificadas detrás de la instrucción. Si se omite el primer dato de LIST, visualizará desde el principio, y si se omite el último dato, visualizará hasta el final. Si no se especifica ninguna línea visualizará todo el programa.
Ejemplos:

LIST -100	Listará desde la primera línea hasta la 100.
LIST 100-	Listará desde la 100 al final.
LIST 120-180	Listará desde la 120 a la 180.
LIST 20	Listará únicamente la línea 20.
LIST	Listará todo el programa.

En número de línea mayor posible es el 65529.

LIST.	Listará la línea en que se ha producido un error.

llamada a código máquina Para efectuar una llamada a una rutina en código máquina se debe especificar, en primer lugar, la dirección de comienzo de la rutina.
Desde BASIC utilizaremos DEF USR.

A continuación se debe ordenar la ejecución de la rutina, para lo que en BASIC emplearemos la instrucción USR.
Se pueden pasar y recibir parámetros de la rutina mediante USR.
Es aconsejable almacenar el programa antes de realizar estas llamadas, para no perderlo en caso de que se produzcan errores no recuperables.

LLIST Imprime en impresora la línea o líneas especificadas detrás de la instrucción. Su funcionamiento es análogo al de la instrucción LIST.
Si se utiliza esta instrucción sin empresora, el ordenador queda parado, y para poder proseguir, habrá que pulsar CTRL+STOP.

LOAD Instrucción que carga un archivo ASCII en la memoria del ordenador desde el dispositivo especificado.

```
LOAD "CAS:PROG1"
```

Cargará un programa desde el cassette previamente guardado con:

```
SAVE "CAS:PROG1"
```

LOC Una función de BASIC de disco que proporciona el número de sectores, tratados hasta el momento de un archivo determinado, el que se abrió con el argumento de LOC.

```
PRINT LOC(3)
```

Dá el número de sectores utilizado hasta el momento por el archivo abierto con el número 3. Multiplicando por 128, obtendremos la longitud en bytes.

LOCATE Instrucción que posiciona el cursor en el punto de coordenadas especificadas.

Solamente es utilizable en las pantallas de texto.

```
LOCATE 15,10:PRINT"HOLA"
```

Imprimirá HOLA en la posición 15,10.
Dispone de una variante para que se visualice el cursor o no:

```
LOCATE 15,10,0    No se visualiza el cursor.
LOCATE 15,10,1    Sí se visualiza el cursor.
```

Ejemplo:

```
10 FOR X=1 TO 30
20 FOR Y=1 TO 20
30 LOCATE X,Y,1:FOR A=1 TO 100:NEXT
40 NEXT Y
50 NEXT X
```

LOG Calcula el valor del logaritmo natural (de base e) de un número dado. Para calcular un logaritmo en otra base ($\log_a b$) puede realizarse el cálculo mediante la expresión numérica:

```
LOG(b)/LOG(a)
```

Ejemplo:

```
PRINT LOG(2.7182818284592)
```

Dá como resultado el valor 1.

logaritmo Es una función numérica que se define como el número al que hay que elevar la base del logaritmo para obtener el argumento de la función.

$$\log_b N = x \quad \text{si} \quad b^x = N$$

LPOS Es una función BASIC que permite conocer la columna en que se encuentra el cabezal de impresión.
Para su utilización precisa una variable fantasma.

```
PRINT LPOS(A)
```

Imprime la posición del cabezal.
Puede no coincidir con la posición real.

LPRINT Instrucción BASIC para la impresión de expresiones en la impresora.
Imprime constantes numéricas o alfanuméricas, como en:

```
LPRINT "HOLA"
```

o las expresiones asignadas a cualquier tipo de variables.
Los separadores actúan de igual forma que en PRINT
El signo ? no sirve como en el caso de PRINT.

LPRINT USING Instrucción BASIC para la impresión de expresiones en la impresora según un formato determinado.
Actúa de forma análoga a PRINT USING.

LSET Instrucción de BASIC de disco empleada en el manejo de archivos de acceso directo.
Transfiere los datos de una variable de cadena a una variable reservada por FIELD, justificándolos a la izquierda.
Para su utilización, hacemos:

```
LSET variable de cadena reservada por FIELD=
variable de cadena a transferir
```

De esta forma, los datos se transfieren al buffer o memoria intermedia. Para transferirse al disco, debe emplearse PUT.

MAC Es una instrucción de ensamblador que especifica el comienzo de una definición de MACRO.
Por ejemplo:

```
Nombre: MAC
        ...
        ...
        definición de MACRO
        ...
        ...
        ENDM
```

macro Es una subrutina en lenguaje ensamblador definida una sola vez y admitida por el programa ensamblador, que permite utilizarla repetidamente haciendo referencia únicamente a su nombre.
Dicha subrutina puede expresar ciertos datos en forma de parámetros que son pasados, con los valores oportunos, en el momento de su utilización.

```
TRANS: MAC
       LD HL,=0
       LD DE,=1
       LD BC,=2
       LDIR
       ENDM
```

Define un macro llamado TRANS que transfiere un cierto número de datos contenido en BC desde su posición en HL y siguientes, hasta DE y siguientes.
Para utilizarlo hacemos, por ejemplo:

```
TRANS 50000,60000,1000
```

o cualesquiera otras expresiones válidas.

manejo de errores El manejo de errores en el MSX-BASIC está encomendado a la instrucción

```
ON ERROR GOTO número de línea
```

que ante un error, bifurca la ejecución al número de línea especificado.

La rutina de tratamiento de errores debe finalizar con RESUME.
ERR toma el código del error producido, y ERL el número de línea donde se produjo.

manejo de interrupciones El manejo de interrupciones en el MSX-BASIC está regulado por instrucciones especialmente dedicadas a esta tarea.
(Véase interrupción).
Las interrupciones del sistema se realizan en modo 1. El gancho situado en &HFD9F controla las interrupciones generadas por el VDP.

manipulación de la RAM de video
La manipulación de la RAM de video es relativamente sencilla si se conoce su estructura, es decir, las tablas en que se divide, lo que contiene cada unda de ellas y sus direcciones de comienzo y longitud.
(Véase tabla de atributos de sprite, tabla de color, tabla de nombres de patrones, tabla de patrones de sprite y tabla de patrones generadores).
Para conocer su contenido y modificarlo se emplean la función VPEEK y la instrucción VPOKE, para sus direcciones la función BASE.
La función VDP nos dá el valor almacenado en los registros de escritura del VDP.

mapa de bits Hace referencia a los gráficos cuyos puntos están unívocamente representados por bits en memoria.

MAXFILES Instrucción BASIC que determina el número X de archivos que pueden abrirse simultaneamente.

X no podrá ser mayor que 15.
En caso de no utilizar la instrucción MAXFILES, sólo podrá estar abierto un único archivo cada vez.
Esta instrucción actúa como una variable (del sistema) a la que se asigna el número de archivos simultáneos.

MAXFILES=5

Reserva espacio de trabajo para 5 archivos simultáneos.

mayúsculas Son las letras cuyo código ASCII está comprendido entre 65 y 90.

memoria Dispositivo que permite el almacenamiento de información.
Existen dos tipos fundamentales:

RAM: memoria de acceso directo de lectura y escritura.

ROM: memoria que sólo permite la lectura.

En la primera se guardan los programas, datos, variables del usuario y del sistema, etc. En la segunda se encuentran el intérprete BASIC y el sistema operativo.
Podemos representarla mediante un mapa de memoria:

MAPA DE MEMORIA MSX-BASIC 64K

Dirección	Contenido
&HFFFF &HF380	Area reservada Ordenador
&HC000	Area de usuario
&H8000 &H4000 &H0000	Intérprete Basic y Sistema Operativo (32K)

memoria de usuario En el MSX-BASIC es la zona de memoria comprendida entre las direcciones &H8000 y &HF380, es decir, 28815 bytes para los sistemas con al menos 32K de RAM.

mensajes de error Son los mensajes que aparecen en pantalla al producirse un error.
A cada uno de ellos le corresponde un código en función del tipo de error.
Son los siguientes:

1. NEXT without FOR: una instrucción NEXT carece de su FOR correlativo.
2. Syntax error: se ha producido un error en la sintaxis.
3. RETURN without GOSUB: una instrucción RETURN carece de su GOSUB previo.
4. Out of DATA: se ha utilizado READ cuando se ha leido la totalidad de DATA'S.
5. Illegal function call: el argumento de la función está fuera de rango.
6. Overflow: se ha excedido la capacidad de cálculo aritmético del ordenador.
7. Out of memory: se ha agotado la memoria disponible.
8. Undefined line number: se ha llamado a una línea no definida.
9. Subscript out of range: el subíndice de una variable de conjunto es demasiado alto o no adecuado.
10. Redimensioned array: se ha tratado de dimensionar una variable de conjunto ya dimensionada.
11. Division by zero: se ha intentado dividir por cero.
12. Illegal direct: se ha ejecutado en modo directo una instrucción que no lo admite.

13. Type mismatch: se han asignado valores a variables de distinto tipo.
14. Out of string space: el espacio para tratamiento de cadenas no ha sido suficiente.
15. String too long: una cadena ha superado los 255 caracteres.
16. String formula too complex: se ha creado una expresión alfanumérica demasiado complicada.
17. Can't continue: no se puede proseguir la ejecución del programa.
18. Undefined user function: no se ha definido la función de usuario a la que se ha llamado.
19. Device I/O error: se ha producido un error en un dispositivo de Entrada/Salida.
20. Verify error: se ha producido un error de verificación en un programa.
21. NO RESUME: falta la instrucción RESUME.
22. RESUME without error: se ha alcanzado una instrucción RESUME sin haberse producido error.
23. Unprintable error: no existe mensaje de error para el error producido.
24. Missing operand: un operador carece de sus respectivos operandos.
25. Line buffer overflow: se está manejando una línea excesivamente larga.

Los códigos de error 26 a 49 no están disponibles ni definidos.

50. FIELD overflow: la longitud de FIELD supera la declarada en OPEN.
51. Internal error: error interno.
52. Bad file number: se hace referencia a un archivo no abierto.
53. File not found: archivo no encontrado.
54. File already open: se intenta abrir un archivo anteriormente abierto.

55. Input past end: se ha sobrepasado el final del archivo.
56. Bad file name: el nombre de archivo es incorrecto.
57. Direct statement in file: ha aparecido una instrucción en modo directo.
58. Sequential I/O only: se ha accedido a un archivo secuencial en modo directo.
59. File not OPEN: se ha accedido a un archivo cerrado.
60. Bad allocation table: el direccionamiento de un disco se encuentra en malas condiciones.
61. Bad file mode: el modo de acceso al archivo es incorrecto.
62. Bad drive name: el nombre de la unidad de disco es incorrecto.
63. File still open: hay que cerrar el archivo con CLOSE.
64. File already exists: el archivo actual existe.
65. Disk full: el disco está completo.
66. Too many files: hay demasiados archivos en el disco.
67. Disk write protected: el disco está protegido.
68. Disk I/O error: se ha producido un error no definido.

Los códigos de error comprendidos entre 70 y 255 están disponibles para su definición por el usuario.

menú Es una lista de opciones presentadas al usuario por el programa.
La elección entre las mismas se realiza mediante un número o letra. Ejemplo:

```
1. INTRODUCIR DATOS.
2. EJECUCION DE CALCULOS.
3. SALIDA DE RESULTADOS.
```



```
PULSE EL NUMERO DE LA OPCION DESEADA.
```

Se puede programar mediante la instrucción:

```
ON ... GOTO, ON ... GOSUB
```

MERGE Instrucción BASIC que carga un programa grabado en formato ASCII (SAVE "CAS:ASUB2"), y lo fusiona con el existente en la memoria del ordenador. En caso de coincidir los números de línea de los programas, prevalecerán las instrucciones del programa fusionado con MERGE.
Ejemplo:

```
MERGE "CAS:ASUB2"
```

por omisión:

```
MERGE "CAS:"
```

grabará el primero que encuentre.
Si el dispositivo es el cassette, podrá omitirse el nombre de dispositivo:

```
MERGE "ASUB2"
```

microprocesador Es un único circuito integrado que contiene la unidad aritmético lógica, la unidad de control, los registros y las salidas de las bases de datos, direcciones y control.
En el sistema MSX el microprocesador empleado es el Z-80 de Zilog, funcionando a una frecuencia de 3.58 MHz.
Existe, asímismo, un procesador independiente para la pantalla, el TMS9129 de Texas Instruments.

**MID$** Función alfanumérica de BASIC que permite obtener un fragmento de una cadena MID$(A$,I,N) de N caracteres a partir del I-ésimo de la cadena A$, contados de izquierda a derecha.

```
A$="SUBMARINO"
PRINT MID$(A$,4,3)
```

Se obtendrá MAR.
I toma valores entre 1 y 255, y N entre 0 y 255, obteniéndose la cadena nula para N=0.
Si después del I-ésimo no existen N caracterres, se obtendrán únicamente los caracteres existentes en A$.
Si el valor de I es mayor que la longitud de la cadena, se obtendrá la cadena nula.
Si se omite N, obtendremos todos los caracteres de A$ a partir del I-ésimo.
Los caracteres gráficos contabilizan como caracteres dobles.

MID$=B$ Instrucción de BASIC que sustituye parte de una cadena en otra.

```
MID$(A$,I,N)=B$
```

Se sustituirán, a partir del I-ésimo caracter de A$, los N primeros caracteres de B$, contando de izquierda a derecha.
La longitud de A$ siempre permanecerá invariable.
I toma valores entre 1 y 255, y N entre 0 y 255.

```
10 A$="HIPOCLORITO"
20 B$="BROMO"
30 PRINT A$
40 MID$(A$,5,4)=B$
50 PRINT A$
RUN
```

Imprimirá:

```
HIPOCLORITO
HIPOBROMITO
```

Si el valor de I es mayor que la longitud de A$ dará error.
Si se omite N, se sustituirán tantos caracteres de B$ como posiciones existan entre el I-ésimo y el último caracter de A$.
Los caracteres gráficos contabilizan como caracteres dobles.

minúsculas Son las letras cuyo código ASCII está comprendido entre 97 y 122.

MKD$ Es una función de BASIC de disco empleada en el manejo de archivos de acceso directo.
Se utiliza para convertir expresiones numéricas de doble precisión en cadenas de 8 bytes de longitud para su almacenamiento en archivos de acceso directo (aleatorio).

```
MKD$(A#)
```

**MKI$** Es una función de BASIC de disco semejante a MKD$. Como argumento utiliza números en precisión entera y retorna cadenas de 2 bytes de longitud.

```
MKI$(A%)
```

**MKS$** Es una función de BASIC de disco semejante a MKD$. Como argumento utiliza números en precisión simple y retorna cadenas de 4 bytes de longitud.

```
MKS$(A!)
```

mnemónico Es una palabra fácil de recordar que representa cada una de las instrucciones del microprocesador.
Supone una gran ventaja sobre la representación binaria de las mismas instrucciones.
En este Diccionario se han recogido las principales instrucciones del procesador, precisamente mediante sus mnemónicos.
Ejemplo:

```
LD A,8
```

LD es abreviatura de LOAD y significa CARGA en el registro A el valor 8.

Por desgracia, una parte de las ventajas de los mnemónicos se pierde para los usuarios de habla hispana.

modem Un dispositivo que permite la transferencia de datos entre dos ordenadores a través de la línea telefónica.
Este dispositivo cobra particular importancia cuando el otro ordenador dispone de una gran base de datos, pues de esta forma, mediante nuestro ordenador doméstico, podemos acceder a dicha información.

modos de direccionamiento
(Véase direccionamiento).

modos de pantalla En el MSX-BASIC existen cuatro modos de pantalla:
- Modo de texto de 40 columnasx24 líneas:
  Dispone solamente de dos colores simultáneos. No admite sprites. Los caracteres son comprimidos a una matriz de 6x8 puntos.
- Modo de texto de 32 columnasx24 líneas:
  Los caracteres pueden tener distintos colores, hasta 16, en grupos de ocho. Admite sprites. La matriz es de 8x8 puntos.
- Modo gráfico de alta resolución con 32 columnasx 24 líneas y coordenadas gráficas de 256x192 puntos:
- Dispone de 16 colores, y la única limitación de que cada segmento horizontal de 8 puntos solamente dispone de dos colores.
- Modo gráfico multicolor:
  Aunque admite coordenadas hasta 256x192 puntos, cada una de ellas, junto con otras tres, representan un único bloque formado por 4x4 puntos.

  Cada bloque puede tomar cualquiera de los 16 colores sin limitación. Los caracteres cuadriplican su tamaño.

Los distintos modos se seleccionan desde BASIC mediante SCREEN.

monitor Es un tubo de rayos catódicos semejante al de televisión que carece, por lo general, del sintonizador, y cuyas características se han mejorado para dar una imagen de mejor calidad y resolución, y producir menos fatiga.
Pueden ser monocromos (blanco/negro, verde/negro, ambar/negro) o de color.

motor Instrucción BASIC para el control del cassette, que conectará (o desconectará) el motor en caso de que esté desconectado (o conectado).
A tal fin, el terminal de control remoto estará conectado al ordenador y la tecla "PLAY" o "RECORD" del cassette pulsada.
También puede usarse como:

MOTOR ON	Conecta
MOTOR OFF	Desconecta

MSX Son las abreviaturas de MicroSoft eXtended, y constituyen una serie de especificaciones técnicas para el desarrollo de un ordenador.
Fueron presentadas en 1.983 por la empresa Microsoft Inc. y comprenden especificaciones de máquina, lenguajes y sistema operativo.
(Véase configuración MSX).

MSX-2 Representa la segunda generación de especificaciones MSX.
Las principales variaciones se producen en gráficos y sonido.

Se utiliza un nuevo VDP (V-9938) capaz de manejar 80 columnas y 512x424 puntos en 16 colores, con un total de 256 colores en resoluciones más bajas. Mejora la representación de sprites y se admite la fusión con imágenes procedentes de videos externos.
La memoria VRAM es de 128K. También se altera el PSG, que pasa a disponer de nueve canales y modulación de frecuencia.
Se mantiene el procesador Z-80 en aras de la compatibilidad con el sistema anterior.
El lenguaje BASIC se amplía con las instrucciones:

- COLOR nº de paleta, intensidad de rojo, de verde, de azul.
  Utilizada para obtener distintas intensidades de color.
- COLOR SPRITE nº de plano, expresión de cadena.
  Para definir el color de sprites línea a línea.
- SET VIDEO
  Para superposición de imágenes.
- COPY VIDEO
  Para digitalizar una señal de video externa.
- GET DATE, GET TIME
  Presentan la fecha y hora del reloj interno (con batería).
- SET DATE, SET TIME
  Determinan la fecha y hora.
- SET PASSWORD
  Define una palabra clave de acceso.
- SET TITLE
  Define un mensaje para visualizar en la reposición.
- SET PROMPT
  Permite cambiar el mensaje Ok.

- SET SCREEN
  Determina las características de pantalla a utilizar.

MSX-DOS Es el nombre del Sistema Operativo de Disco (Disk Operative System) de la especificación MSX.
(Véase sistema operativo de disco).
Existen dos versiones: en cartucho y en disco.
La primera opera forzosamente en BASIC, al que se añaden las instrucciones de manejo de discos.
La segunda es semejante a las de otros ordenadores (como el IBM PC y compatibles) y dispone de numerosas instrucciones de sistema.
Puede operar con otros lenguajes (existen versiones disponibles de ensamblador, Pascal y lenguaje C).
Además de este sistema operativo, los ordenadores MSX también admiten la versión 2.2 en disco de CP/M.

música El generador de sonido del sistema MSX puede ejecutar tres canales de sonido simultáneos en una emplitud de 8 octavas.
A través de las instrucciones SOUND, PLAY se pueden modificar los distintos controles de tono, duración, volumen, envolvente, modulación, etc.

NAME Una instrucción de BASIC de disco que permite cambiar el nombre de un archivo.
En primer lugar situamos el nombre actual, y a continuación, el nombre nuevo.

```
NAME "PROG1.BAS" AS "PROG2.BAS"
```

Sustituye el nombre PROG1 por PROG2.

NEG Mnemónico de instrucción Z-80 que calcula el complemento de dos del contenido del acumulador.

nemónico
(Véase mnemónico).

NEW Instrucción que borra todo programa BASIC de la memoria, así como todas sus variables.
Los programas en código máquina permanecen invariables, siempre que para su almacenamiento se haya reservado espacio mediante CLEAR.

notación binaria La notación binaria (en base 2) utiliza solamente los dígitos 0 y 1 para representar las distintas cantidades.
Se diferencia sustancialmente de la notación decimal, que es la normalmente utilizada, y que precisa de los dígitos 0, 1, 2 ... 9.
Sin embargo, las reglas que subyacen a la utilización de una u otra notación son siempre las mismas y se fundan en los principios de la notación posicional. (Véase notación posicional).
Como aproximación a la notación binaria, se muestra la secuencia siguiente:

Decimal	Binario
0	0
1	1

2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

Para conocer el valor decimal de un número binario operamos en la siguiente forma:

$$11010011 = 1x2^7 + 1x2^6 + 0x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 =$$

$$= 1x128 + 1x64 + 1x16 + 1x2 + 1x1 =$$

$$= 128 + 64 + 16 + 2 + 1 = 211$$

que es el equivalente decimal del número binario:

11010011

Para conocer el valor binario de un número decimal lo dividimos repetidamente por 2:

```
211|2
011 105|2
  1  05 52 |2
      1 12  26 |2
         0   0  13 |2
                 1  6  |2
                    0   3  |2
                        1   1
```


Los sucesivos restos y el último cociente en orden inverso es el equivalente binario del número:

11010011

El ordenador admite números binarios si van precedidos de &B.

notación científica La notación científica consta de una mantisa que dá la precisión del número y un exponente que determina su tamaño.
En el ordenador, el exponente se separa con la letra D en doble precisión, o con la letra E en precisión simple.
Así:

12.34E10 expresa el nº 123400000000

La notación científica normalizada utiliza una única cifra entera. En el ejemplo anterior, sería

1.234E11

notación decimal La notación decimal (en base 10) utiliza los dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, y 9, siendo la normalmente utilizada.
Cada posición significa una determinada potencia de diez. Los números a la derecha representan unidades, la siguiente columna a la izquierda decenas, centenas, millares, y así, sucesivamente.
Ejemplo:

$$234 = 2x10^2 + 3x10^1 + 4x10^0 = 200 + 30 + 4 =$$

$$= 234$$

notación de complemento de dos
Es un método de codificación de números negativos en notación binaria.
Se consideran negativos los valores cuyo primer bit (más significativo) es 1.

Para traducir un valor numérico negativo a esta notación se calcula el equivalente binario del número como si fuera positivo. El valor obtenido se invierte bit a bit (los valores 1 se hacen 0 y viceversa).
Por último, se le suma 1.
Ejemplo: Sea el número -100; el equivalente binario de 100 es 01100100, que invertido resulta 10011011, y sumando 1, obtenemos 10011100.
Si utilizamos dos bytes, podemos representar entre -32768 y 32767; con un único byte tendremos desde -128 a 127.

notación hexadecimal La notación hexadecimal (en base 16) utiliza, además de los dígitos habituales 0, 1, 2 ... 9, los representados por las letras A, B, C, D, E y F, con las siguientes equivalencias:

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

Siendo, por tanto, el número total de "dígitos" de 16, base de esta notación.
Como aproximación a la notación hexadecimal, se muestra la secuencia siguiente:

Decimal	Hexadecimal
0	0
1	1
....	....
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
....	....
25	19
26	1A
27	1B
....	....
31	1F
32	20

Cada cifra de un número hexadecimal corresponde a cuatro de uno binario:

A	F
1010	1111

Para obtener el valor decimal de un número hexadecimal es suficiente con operar:

$$A1F2 = Ax16^3 + 1x16^2 + Fx16^1 + 2x16^0 =$$
$$= 10x16^3 + 1x16^2 + 15x16^1 + 2x16^0 =$$
$$= 10x4096 + 1x256 + 15x16 + 2x1 =$$
$$= 40960 + 256 . 240 + 2 =$$
$$= 41458$$

Teniendo en cuenta las equivalencias de las letras.
Para obtener el equivalente hexadecimal de un número decimal, lo dividimos repetidamente por la base:

```
41458 |16
 94    2591 |16
 145    99   161 |16
  018   031   01   10
    2    15
```

Con los restos y último cociente en orden inverso formamos el número:

10, 1, 15, 2 A1F2

El ordenador admite números hexadecimales, si van precedidos de &H.

notación octal La notación octal (en base 8) utiliza solamente los dígitos 0, 1, 2, 3, 4, 5, 6 y 7 para representar las distintas cantidades.

Como aproximación a esta notación, se muestra la secuencia siguiente:

Decimal	Octal
0	0
1	1
2	2
....	
7	7
8	10
9	11
....	
15	17
16	20

Para conocer el valor decimal de un número octal, operamos de la siguiente forma:

$$4527 = 4x8^3 + 5x8^2 + 2x8^1 + 7x8^0 =$$
$$= 4x512 + 5x64 + 2x8 + 7 =$$
$$= 2048 + 320 + 16 + 7 = 2391$$

Para obtener el equivalente decimal de un número octal, lo dividimos repetidamente por la base:

```
2391 |8
 79   298 |8
  71   58  37 |8
   7    2   5  4
```

Con los restos y último cociente en orden inversos formamos el número 4527.

El ordenador admite números octales si van precedidos por &O.

notación posicional Cuando manejamos un número con varias cifras, cada una representa un determinado valor que depende de su posición dentro del número.

Sea, por ejemplo, el número: 234

El número situado a la derecha representa las unidades, en este caso, cuatro unidades:

$$4x\ 10^0 = 4x1 = 4$$

El siguiente representa las decenas, es decir, tres decenas o:

$$3x10$$

El siguiente las centenas, en este caso, dos centenas, o lo que es igual:

$$2x10x10 = 2x10^2$$

Por lo tanto, cada cifra representa un valor que depende de su posición en el número, y que es igual a la cifra por la base de numeración elevada a un exponente, o si la posición que ocupa es de unidades, 1 si es la siguiente por la izquierda (decenas cuando operamos con base 10), 2 si es la tercera por la izquierda, y así, sucesivamente.

En nuestro ejemplo:

$$234 = 2x10^2 + 3x10^1 + 4x10^0 =$$
$$= 200 + 30 + 4 = 234$$

Esta condición se cumple para cualquier número en cualquier base.
En base binaria (2) tenemos:

$$11010011 = 1x2^7 + 1x2^6 + 0x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 =$$
$$= 1x128 + 1x64 + 1x16 + 1x2 + 1x1 =$$
$$= 128 + 64 + 16 + 2 + 1 = 211$$

equivalente decimal del número binario 11010011.
En base hexadecimal (16) sería:

$$A1F2 = Ax16^3 + 1x16^2 + Fx16^1 + 2x16^0 =$$
$$= 10x16^3 + 1x16^2 + 15x16^1 + 2x16^0 =$$
$$= 10x4096 + 1x256 + 15x16 + 2x1 =$$
$$= 40960 + 256 + 240 + 2 = 41458$$

equivalente decimal del número hexadecimal A1F2.

OCT$ Función que dá la expresión octal de datos numéricos en forma de datos alfanuméricos.

```
A$ = OCT$(8)
PRINT A$
```

Resulta:

```
10
```

ON ERROR GOTO Instrucción BASIC que bifurca el flujo del programa a la línea especificada cuando se encuentre con un error.
Veamos un ejemplo:

```
10 ON ERROR GOTO 100
20 INPUT L
30 M=SQR(L)
40 N=1/M
50 PRINT N
60 END
100 PRINT "ERROR TIPO";ERR;"EN LA LINEA";ERL
110 RESUME 20
```

Para anular la instrucción:

```
ON ERROR GOTO
```

ejecutar:

```
ON ERROR GOTO 0
```

ON...GOSUB Instrucción BASIC que bifurca el flujo del programa a una de las subrutinas que se inician con el número de línea especificado, dependiendo del valor de la expresión.

```
ON Z GOSUB 100,150,200
```

Así, para:

$1 \leq Z < 2$	Bifurcará a la subrutina que empieza por 100.
$2 \leq Z < 3$	Bifurcará a la subrutina que empieza por 150.
$3 \leq Z < 4$	Bifurcará a la subrutina que empieza por 200.

Para:

0≤Z<1 y 3<z	El flujo irá a la siguiente sentencia ON ... GOSUB.
Z<0 y 255<Z	Dará error.

La subrutina ha de terminar con una instrucción RETURN que devolverá el flujo a la instrucción siguiente a ON ... GOSUB.

ON...GOTO Instrucción BASIC que bifurca el flujo del programa a los números de línea especificados, dependiendo del valor que tome la expresión:

```
ON Z GOTO 50,100,150
```

Los intervalos de la expresión (Z), así como sus respuestas, son análogos a los de la instrucción ON ... GOSUB.

ON INTERVAL GOSUB Bifurca el flujo del programa mediante interrupción a la subrutina especificada después de haber transcurrido el período de tiempo especificado.

```
10 ON INTERVAL=250 GOSUB 150
20 INTERVAL ON
```

ON KEY GOSUB Bifurca el flujo del programa, mediante una interrupción producida al pulsar una tecla de función, a la subrutina especificada por su número de línea. Continuando por su cauce normal tras haberse encontrado con un RETURN como fin de subrutina.
Solamente podrán utilizarse las cinco primeras teclas de función.

```
10 ON KEY GOSUB 50,60,70
20 KEY(1)ON:KEY(2)ON:KEY(3)ON
30 FOR I=1 TO 3000:LOCATE 8,5: PRINTI: NEXTI
40 END
50 PRINT "A":RETURN
60 PRINT "B":RETURN
70 PRINT "C":RETURN
```


La selección se hace secuencialmente, yendo los números de línea de las subrutinas separados por ","
(Véase interrupción).

ON SPRITE GOSUB Bifurca el flujo del programa mediante una interrupción, al solaparse patrones de figura móvil, a la subrutina especificada.

```
10 ON SPRITE GOSUB 100
20 SPRITE ON
```

(Véase interrupción).

ON STOP GOSUB Bifurca el flujo del programa mediante una interrupción a la subrutina especificada cuando se pulsan las teclas:

CTRL + STOP

Ejemplo:

```
10 ON STOP GOSUB 200
20 STOP ON
```

Debe usarse con cuidado, ya que podemos perder el control al ser ésta la única instrucción que no devuelve al modo directo.

ON STRIG GOSUB Bifurca el programa mediante una interrupción a la subrutina especificada cuando se pulsa la barra espaciadora o uno de los dos disparadores de los dos joystick.
La selección se hace secuencialmente, yendo los números de línea de las subrutinas separados por ","

El orden secuencial es el siguiente:
- Barra espaciadora.
- Joystick A, disparador 1.
- Joystick B, disparador 1.
- Joystick A, disparador 2.
- Joystick B, disparador 2.

(Véase interrupción).

OPEN Instrucción para la apertura de un archivo en el modo escritura/lectura en el dispositivo especificado.

```
OPEN "CAS:DATOS1"FOR OUTPUT AS #1
```

Abre un archivo en el cassette con el nombre "DATOS1" para escritura por el canal 1.
Los dispositivos son:

CAS:	cassette
CRT:	pantalla en el modo de texto
GRP:	pantalla en el modo gráfico
LPT:	impresora
CAT:	cartucho (en caso de tenerlo)

En el nombre solamente se considerarán los seis primeros caracteres.
Tenemos dos modos:

OUTPUT:	escritura secuencial en el archivo.
INPUT:	lectura secuencial del archivo.

El número de canal a utilizar ha de ser menor o igual al número especificado por MAXFILES, y en cualquier caso, mayor o igual que 1.
En el BASIC de disco se admiten los dispositivos A:, B:, que corresponden a las unidades de disco. Asímismo, los modos se amplían con APPEND, que permite añadir datos a un archivo secuencial existente.

El nombre de archivo se amplía con la extensión, que indica la procedencia o tipo de archivo. Consta de tres letras y lleva un punto intercalado.

```
OPEN "A:DATOS.DAT" FOR APPEND AS #1
```

Abre un archivo de datos en la unidad de discos A para ampliación por el canal 1.
Para la utilización de archivos de acceso directo en disco, la sintaxis de OPEN se altera. En este tipo de archivos, no es preciso especificar entrada o salida, al poderse realizar ambas operaciones conjuntamente. Por el contrario, debe especificarse la longitud total del registro, para acomodarla a nuestras necesidades, pues si se omite, adoptará el valor 128.

```
OPEN "A:DATOS.DAT" AS #1 LEN=64
```

Abre un archivo de datos en la unidad de discos A de acceso directo, con una longitud total de registro de 64 caracteres.

operación Es una acción ejecutada por el ordenador que, generalmente, relaciona dos expresiones (numéricas, alfanuméricas o lógicas) para obtener un valor de cualquiera de los tipos.
Pueden ser:
- Operaciones aritméticas.
- Operaciones de relación.
- Operaciones lógicas.
- Operación concatenación.

operación aritmética Relaciona dos operandos numéricos (expresiones numéricas) y proporciona un valor también numérico.
Existen distintos tipos de operaciones aritméticas que se representan por sus respectivos operadores. Son las siguientes:

^ Potenciación
\- Cambio de signo(operador unario)
*,/ Multiplicación, división
\ División entera
MOD Módulo (resto entero de división entera)
+,- Suma, resta

Si se opera sin paréntesis, las operaciones se ejecutan de izquierda a derecha y en el orden de la lista anterior.
La mayoría de ellas son suficientemente conocidas.
La división entera reduce a enteros el dividendo y divisor, seguidamente ejectúa la división y reduce a entero el cociente. Ejemplo:

3\4 = 0
5\4 = 1

La operación módulo nos dá el resto entero de la división entera. Ejemplo:

3 MOD 4 = 3
5 MOD 4 = 1

operación concatenación Realiza la unión de dos cadenas alfanuméricas:

"ORDEN" + "ADOR MSX"

equivale a:

"ORDENADOR MSX"

operación de relación Compara dos operandos numéricos o alfanuméricos y proporciona un valor lógico (verdadero o falso).
Existen distintos tipos de operaciones de relación que se representan por sus respectivos operadores.
Son los siguientes:

= Igualdad
< Menor que
> Mayor que

<> >< Desigualdad
≤ Menor o igual que
≥ Mayor o igual que

Al producir un valor lógico, una operación de relación puede ser operando de una operación lógica.
Ejemplos:

A>0

es verdadero si A es mayor que 0 y falso si A es igual a 0, ó A es menor que 0.

A>0 AND B=1

Es una operación lógica que utiliza como operandos dos operadores de relación.

operación lógica Relaciona dos expresiones lógicas para obtener un resultado lógico (verdadero o falso). También relaciona dos expresiones numéricas enteras para calcular un resultado numérico entero.
En el primer caso, el resultado obtenido depende de la "Tabla de Verdad" de cada operación.
Existen distintos tipos de operaciones lógicas que se representan por sus respectivos operadores.
Son las siguientes:

- Negación NOT: Es una operación unaria, pues solamente precisa de un único operando.
  Si A es verdadero, NOT A es falso, y viceversa.
  Su Tabla de Verdad es:

X	NOT X
V	F
F	V

- Producto lógico AND: Es, al igual que las siguientes, una operación binaria, al precisar de dos operandos.
  Para ser verdadera, los dos operandos tienen que serlo simultáneamente:

X	Y	X AND Y
V	V	V
V	F	F
F	V	F
F	F	F

- Suma lógica OR: Para ser verdadera es suficiente que al menos uno de los dos operandos lo sea:

X	Y	X OR Y
V	V	V
V	F	V
F	V	V
F	F	F

- Suma lógica exclusiva XOR: Su Tabla de Verdad es

X	Y	X XOR Y
V	V	F
V	F	V
F	V	V
F	F	F

- Equivalencia EQV: Es la negación de la suma lógica exclusiva.
  Su Tabla de Verdad es:

X	Y	X EQV Y
V	V	V
V	F	F
F	V	F
F	F	V

- Implicación IMP: Su Tabla de Verdad es:

X	Y	X IMP Y
V	V	V
V	F	F
F	V	V
F	F	V

En el caso de operar con expresiones numéricas de tipo entero, el proceso seguido por el ordenador puede dividirse en las siguientes fases:

1º El ordenador convierte los operandos a su equivalente binario.
2º Se aplica dígito a dígito la operación señalada formando el resultado en notación binaria.
3º Se calcula el equivalente decimal de dicho resultado.

Veámoslo con un ejemplo:

249 AND 29

Aplicando la primera fase, obtenemos:

11111001 AND 00011101

Para la segunda fase, operamos bit a bit:

1 AND 0 = 0
1 AND 0 = 0
1 AND 0 = 0
1 AND 1 = 1
1 AND 1 = 1
0 AND 1 = 0
0 AND 0 = 0
1 AND 0 = 0

Para realizar esta operación, hemos supuesto que verdadero = 1, y falso=0

Por último, recomponemos el valor decimal:

00011000 = 24

operador Es un símbolo o palabra que representa una determinada operación.
En algunos casos, como +, pueden representar dos operaciones diferentes en función del contexto en que se usen (Adición y Concatenación).
(Véase operaciones ...).

operando Es cada una de las expresiones numéricas, alfanuméricas y lógicas relacionadas por un determinado operador.
Los operadores se clasifican en función de sus operandos.

OR (1) Un operador lógico binario de BASIC que utiliza como operandos expresiones lógicas o numéricas. En el primer caso, se emplea la siguiente Tabla de Verdad:

X	Y	X OR Y
V	V	V
V	F	V
F	V	V
F	F	F

En el segundo caso, se opera en la forma expuesta en operaciones lógicas. Se utiliza al igual que AND para combinar condiciones dentro de una sentencia IF. Siempre que una de ellas se cumpla, se ejecutará la instrucción que sigue a THEN.

(2) Un mnemónico de instrucción Z-80 que ejecuta la operación lógica OR, en forma similar a BASIC pero tomando como operandos un dato, un registro o el contenido de una posición de memoria y el acumulador. Almacena el resultado en A.
Se utiliza al igual que AND para alterar determinados bits de un valor al operarlos con sus respectivos bits de valor 1.

ordenación Consiste en modificar las posiciones relativas de una serie de objetos para que cumplan un determinado orden (numérico, alfabético ...).
Existen diversas técnicas de programación para conseguirlo, una de las cuales es la ordenación burbuja, que se muestra a continuación.
El siguiente programa genera 20 números aleatorios que guarda en una variable de conjunto, los muestra desordenados y después ordenados.


```
10 DIM NA(20):CLS:R=RND(-TIME)
20 FOR N=1 TO 20
30 NA(N)=RND(1)
40 PRINT NA(N)
50 NEXT N
60 F=0
70 FOR N=1 TO 19
80 IF NA(N)<=NA(N+1) THEN 130
90 X=NA(N)
100 NA(N)=NA(N+1)
110 NA(N+1)=X
120 F=1
130 NEXT N
140 IF F<>0 THEN 60
150 CLS
160 FOR N=1 TO 20
170 PRINT NA(N)
180 NEXT N
```

ORG Una instrucción de ensamblador que señala mediante la expresión que le sigue cuál debe ser el origen en la ubicación del programa ensamblado.

organización de la memoria El procesador Z-80 puede direccionar 64K de memoria simultaneamente. En el sistema MSX, estos 64K están divididos en páginas de 16K, cada una de las cuales puede proceder de cuatro ranuras o bloques de 64K. Por lo tanto, aunque simultáneamente solamente puedan manejarse 64K, la memoria total puede ser de hasta 256K.
El PPI (Interfaz Programable para Periféricos) es el encargado de determinar la ranura de la que se obtendría cada página de 16K a través de su puerto A.

A su vez, cada ranura (primaria) puede estar formada por páginas procedentes de otras cuatro ranuras (secundarias), elevándose la memoria total hasta 1 Mbyte.

OTDR Mnemónico de instrucción Z-80 que envía al puerto almacenado en C el contenido de memoria direccionado por HL, disminuyendo HL y B en una unidad y repitiendo hasta que B sea cero.

OTIR Mnemónico de instrucción Z-80 que envía al puerto almacenado en C el contenido de memoria direccionado por HL, aumentando HL y disminuyendo B en una unidad y repitiendo hasta que B sea cero.

OUT (1) Instrucción BASIC que manda un valor comprendido entre 0 y 255 a una de las vías de acceso (direcciones del puerto de E/S utilizadas por el sistema).
El siguiente programa genera un tono a partir del generador de sonido del PPI.

```
10 OUT &HAA,128:OUT &HAA,0
20 GOTO 10
```

(2) Mnemónico de instrucción Z-80 que envía el contenido del acumulador u otros registros al puerto expresamente designado o contenido en C.

OUTD Mnemónico de instrucción Z-80 que envía al puerto almacenado en C el contenido de memoria direccionado por HL, disminuyendo HL y B en una unidad.

OUTI Mnemónico de instrucción Z-80 que envía al puerto almacenado en C el contenido de memoria direccionado por HL y disminuyendo B en una unidad.

PAD Una función BASIC que realiza la lectura de un tablero digitalizador.
Utiliza como argumento los valores 0-7, con el siguiente significado:

Argumento	Significado
0-4	0 no se ha tocado el tablero. -1 Si se ha tocado.
1,5	Coordenada X.
2,6	Coordenada Y.
3,7	0 no se ha pulsado el interruptor.
-1	Si se ha pulsado.

PAINT Instrucción que colorea el área delimitada por una línea cerrada en las pantallas de gráficos.
Han de especificarse las coordenadas de un punto perteneciente al interior o al exterior de la línea cerrada, según la zona que se quiera colorear.
Así:

```
PAINT (125,85),15,1
```

Coloreará de blanco (15) el área encerrada por la línea negra (1) al que pertenece el punto de coordenadas 125,85.
El color de relleno y el color de la línea fronteriza se pueden omitir en base a las siguientes indicaciones.
En SCREEN 3, salvo en el caso de que la línea cerrada y trazada previamente a la ejecución de PAINT tenga el mismo color que el especificado para relleno mediante PAINT, ha de darse el color de la línea cerrada para que el BASIC MSX reconozca la zona a colorear. De esta forma se obtendrán áreas coloreadas y delimitadas por un color dis-

tinto. En caso contrario, se coloreará toda la pantalla.
Ejemplo:

```
10 SCREEN 3
20 CIRCLE (125,85),50,11
30 PAINT (125,85),1,11
40 GOTO 40
```

En SCREEN 2 es condición necesaria que el color de relleno coincida con el de la línea fronteriza. En caso contrario, se coloreará toda la pantalla. En esta pantalla el color de la línea fronteriza en PAINT es inoperante.

```
10 SCREEN 2
20 CIRCLE (125,85),50,11
30 PAINT (125,85),11,255
40 GOTO 40
```

Se admite la variante PAINT STEP(X,Y). (Véase STEP).

pantalla Tubo de rayos catódicos que permite visualizar imágenes.
(Véase monitor).

paralelo Protocolo de comunicaciones (Paralelo-Centronics) que consiste en enviar cada bit separadamente por un conductor precisándose 8 conductores para la transmisión de la información y tres más para control y tierra.

parámetro (1) Valor que hay que añadir a una instrucción para constituir una sentencia.
Puede ser numérico, alfanumérico o lógico.

(2) Valor que se transmite desde BASIC a una rutina de código máquina, a través de la función USR.

paréntesis El paréntesis es un signo especial que altera el orden de ejecución normal (precedencia) de las operaciones.

```
PRINT 2*3+4
```

Imprime el valor 10, mientras que:

```
PRINT 2*(3+4)
```

Imprime el valor 14, al ejecutarse inicialmente la suma 3+4.
Todos los paréntesis abiertos en una expresión deben cerrarse.
Si no se está seguro del orden de ejecución normal debe asegurarse mediante paréntesis.

paridad (1) Es el carácter de la suma de los bits que representan un determinado valor.
Puede ser par e impar.
Se utiliza en eliminación de errores.

(2) Uno de los bits del registro de banderas.

PDL Una función BASIC que realiza la lectura de una raqueta y proporciona un valòr comprendido entre 0 y 255, en función de su posición.
Se admiten hasta 12 raquetas que se diferencian por el argumento de la función (entre 1 y 12).
Los argumentos impares leen el conector A y los pares el B.

PEEK Función que proporciona el contenido de la dirección especificada de la memoria.

```
P=PEEK(1000)
```

Le asigna a la variable P el contenido de la dirección 1000 de la memoria.
La dirección ha de estar comprendida entre -32768 y 65535.

periférico Cualquier dispositivo extremo conectado con el ordenador.
(Véase dispositivo).

pila (1) Es una estructura de datos del tipo "primero en entrar, primero en salir".
Se utiliza para el almacenamiento temporal de datos en instrucciones del tipo GOSUB, etc.

(2) Existe una pila en lenguaje ensamblador, controlada por las instrucciones PUSH y POP, que permite la salvaguarda de registros, direccionamiento, etc.

pixel Cada uno de los puntos luminosos que componen la pantalla.
Sirven para generar el texto y los gráficos.

PLAY Instrucción para la generación de sonidos.
Se dispone de tres canales que pueden funcionar por separado, o simultaneamente. Ocho octavas con sus doce correspondientes semitonos (y sus ocho notas), por tanto, 96 semitonos.

PLAY "n45"

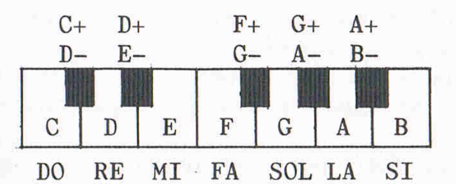
Tocará la nota 45 por el primer canal.

PLAY "CDEF","","DEFG"

Tocará las notas DO, RE, MI, FA por el canal 1, y RE, MI, FA, SOL por canal 3, simultáneamente.
La instrucción PLAY atiende a las siguientes subinstrucciones:

Nn: Especifica una nota dentro de la escala de los 96 semitonos, NO es un silencio.
Otra forma de especificar una nota es fijando previamente la octava.

On: Determina una de las 8 octavas, siendo el valor de inicialización 04.
Las notas, dentro de cada octava, vendrán definidas por las letras de la A a la G. Los bemoles y sostenidos (sólo los de piano) se producen con los signos # ó + para los sostenidos y - para los bemoles, colocados inmediatamente detrás de la nota. La disposición queda como sigue:



DO RE MI FA SOL LA SI

PLAY "O5CDECDEGFEDC"

Tocará las notas correspondientes a la quinta octava.

Vn: Determina el volumen variando entre 0 y 15 como máximo volumen. El valor de inicialización es V8.
La duración de las notas se puede hacer de varias formas. Así, un "." inmediatamente después de la nota, aumenta el 50% la longitud de la nota, y si no, tenemos la subinstrucción siguiente: Ln.

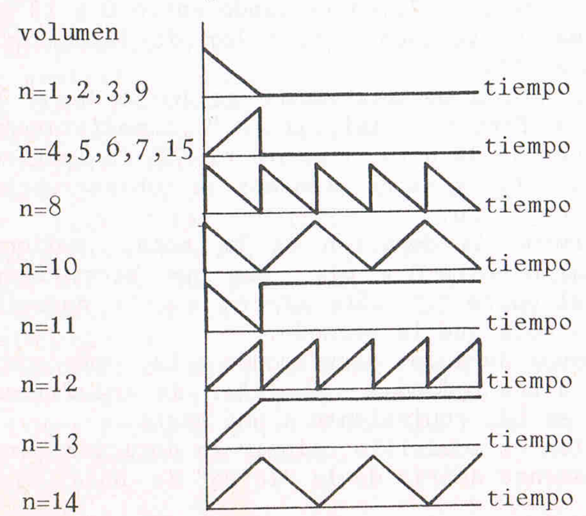
Ln: Determina la duración de la nota, pudiendo variar n entre 0 y 64. Hay que hacer notar que el punto "." sólo afecta a la duración de la nota que le precede.
La mayor duración corresponde a L1, que equivale a una redonda. El valor de inicialización es L4, equivalente a una negra.
También es admisible colocar la duración inmediatamente detrás de la "letra" de nota, omi-

tiendo el caracter "L".
A1 F53
De esta forma, la correspondiente duración sólo afectará a la fijada nota.

Tn: Define la velocidad de ejecución, especificando el número de negras tocadas por minuto. Puede tomar valores entre 32 y 255, siendo el valor de inicialización "T120".
Combinando el tiempo T con la duración L podemos obtener notas más largas o más cortas, que utilizándolos por separado.

Rn: Se utiliza para producir pausas, especifica un silencio. Al igual que Ln, varía entre 1 y 64.
La instrucción PLAY permite definir la forma o patrón de la envolvente del volumen mediante Sn y el número de ciclos a ejecutar de la envolvente por nota mediante Mn.

Sn: Determina el patrón de envolvente del volumen:



El valor de inicialización es S1.

Mn: Determina el número de ciclos de la envolvente que hay en la duración de una nota; n puede tomar valores entre 1 y 65535.
El valor de inicialización es M255.
Cuanto menor sea n, mayor será el número de ciclos por nota.
PLAY "S8M3485CCCD"
Esta disposición nos realizará un único ciclo completo del patrón 8, si el resto de parámetros están inicializados.
PLAY "S8M19000 3CDECDEGFEDC"

Para utilizar variables, han de utilizarse las siguientes variantes:

- Sustituir la cadena por una variable para un canal:
  PLAY A$,B$,C$
- Sustituir parte de la cadena por una variable, se le antepondrá una X y al final ;
  A$="CFD"
  PLAY "DXA$;"
- Introducir una variable numérica en la cadena:
  PLAY "O=I;DCFD"
- Para asignar a una subinstrucción una variable utilizaremos el signo = entre ellos, y terminaremos con ;

El siguiente programa ejecuta un popular villancico:

```
10 A$="t240o4eeeeeeeeegcdeeeefffffeeeeddeddd
   deeeeeeeeegcdeeeefffffeeeggfdcccc"
20 B$="t240o3ccccccccccccccccccccccco2bbbbb
   bbbo3ccccccccccccccccccccccco2bbbbo3cccc
   "
30 C$="t240o2ggggggggggggggggaaaagggggggggg
   ggggggggggggggggaaaaggggggggggg"
40 PLAY A$,B$,C$
```

PLAY n Función que comprueba los canales dando -1 si el canal está activo, y cero en caso contrario; n puede valer 0, 1, 2, 3.
Para n=1,2,3 nos comprueba el canal correspondiente, y para n=0 los comprueba los tres dando como resultado la suma lógica OR, por tanto, si uno de ellos, dos o los tres están activos, obtendremos -1.

```
10 PLAY "","CDEEDC"
20 LOCATE 2,2:PRINT PLAY (2):GOTO 20
```

POINT Función que dá el código de color del punto de coordenadas (X,Y).
Para las zonas no comprendidas en el área 255x191, se obtendrá el valor -1.
Los valores de X e Y pueden estar comprendidos entre -32768 y 32767.

```
P=POINT(127,96)
```

Asigna a P el color del centro de la pantalla.

POKE Instrucción para la escritura en una dirección especificada de la memoria.
La dirección estará comprendida entre -32768 y 65535, y el dato entre cero y 255.

```
POKE 40000,200
```

Asigna a la posición 40000 el valor 200.

POP Mnemónico de instrucción Z-80 que extrae de la pila los dos últimos valores y los carga en un par de registros.
La instrucción complementaria es PUSH.
Se utiliza para salvaguardar valores de registros y poder utilizar estos últimos en otras operaciones.

POS Función que dá la coordenada X de la posición del cursor en los modos de texto.
Ha de utilizarse un argumento arbitrario: POS(0)

```
10 CLS
20 LOCATE 4,9:PRINT POS(0)
```

Imprime la coordenada X del cursor.

PPI Abreviatura en ingles de Interfaz Programable para Periféricos.
(Véase Interfaz Programable para Periféricos).

precedencia Es el orden en que se ejecutan las distintas operaciones.
En el caso de las aritméticas, es el siguiente:

^	Potenciación
-	Cambio de signo
*,/	Multiplicación, división
\	División entera
MOD	Módulo
+,-	Suma, resta

El ordenador explora la expresión de izquierda a derecha hasta encontrar el operador de mayor prioridad, ejecutándolo, y así sucesivamente.
Si se encuentra dos operadores de la misma prioridad, ejecuta el situado a la izquierda.

precisión Es la aproximación con que un valor numérico representa la realidad.
El ordenador dispone de tres gamas de precisión: entera, simple y doble, y opera, si no se le indica lo contrario, en precisión doble.
Los valores de precisión crecientes precisan de un espacio de almacenamiento mayor y de mayores tiempos de ejecución.
Así, la precisión entera necesita 2 bytes, la simple 4 bytes y la doble 8 bytes.

La precisión de valores y variables se puede declarar mediante signos %, !, # ó instrucciones como DEFINT, etc.

PRESET Instrucción que sitúa un punto en las coordenadas especificadas en las pantallas de gráficos. Si se omite el color, tomará el color de fondo, función análoga a la de PSET.
Sirve para realizar la función de LOCATE en las pantallas de gráficos.

PRINT Instrucción para la visualización de cadenas y variables en la pantalla.

Cadenas:	PRINT "HOLA"
	PRINT "HOLA";
Variables:	PRINT A,A$;B$
Caracteres de control:	PRINT CHR$(11)+A$

donde CHR$(11) es equivalente a LOCATE 0,0.

El separador "," avanza a la próxima zona de tabulación, que tienen una anchura de 14 caracteres. El ";" hará que los siguientes datos se encuentren justo a continuación de los primeros, excepto cuando se trabaje con variables numéricas en que se dejara un espacio vacío en caso de ser positivo el número, y el espacio para el signo en caso de ser negativo.
Si después de la instrucción no se pone "," ni ";" se produce una alimentación de línea.
La instrucción PRINT se puede sustituir por el caracter "?"

PRINT USING
Instrucción para la visualización de cadenas y variables en la pantalla, según un formato especificado:

!: Produce la impresión del primer caracter de la cadena o variable.
PRINT USING "!";"HOLA"

\ \: Especifica el número de caracteres que se van a imprimir, aumentados en dos.
PRINT USING "\ \";A$,B$
Si la cadena es más pequeña que el tamaño especificado, se rellena con espacios.

&: Permite insertar una cadena o variable en otra cadena.
A$="BASIC"
PRINT USING "&MSX.";A$
Obtendremos:BASIC MSX.
Si se define más de una cadena o variables, esta secuencia se realizará para todas ellas:
B$="LENGUAJE"
PRINT USING "&MSX.";A$,B$
Obtendremos: BASIC MSX. LENGUAJE MSX.

#: Se utiliza para el formateado de datos numéricos como opción conjunta con el carácter de punto decimal (.) y cualquier cadena que se desee imprimir.

Si el valor no tiene la suficiente longitud, se completará con espacios en blanco en la parte entera y con ceros en la parte decimal.
PRINT USING "###.##";2.3
Se obtendrá: 2.30
Si el valor tiene mayor longitud que el espacio reservado, se obtendrá el caracter % delante del número, si es en el parte entera, o se redondeará, si es en la parte decimal.
PRINT USING "##.##";666.777
Obtendremos: %666.78

El espacio reservado mediante # ha de ser menor que 25 caracteres.
Para la separación de varios datos, se formateará con la ayuda de espacios en blanco.
PRINT USING " ##.## ##.## ##";89,5.777,45.8
Se obtendrá: 89.00 5.78 46
El caracter # se puede utilizar en combinación con los siguientes caracteres: **, ££, +, -, ^^^^ y coma.

**: El doble asterisco situado delante de los caracteres # hace que se rellenen los espacios iniciales con *. Cada * se contará como un dígito en el formato.
PRINT USING "**###.##";34.777
Se obtendrá: ***34.78

££: La doble libra hace que se imprima un sólo caracter libra justo delante del número. Cada libra contará como un dígito en el formato.
PRINT USING "££##.##";3567.777
Se obtendrá: %£3567.78

##£: Añade una £ justo delante del número, rellenando los demás espacios anteriores con *.

+: Añadido delante (o detrás) de los caracteres numéricos (#) en el formato, hace que aparezca el signo delante (o detrás).

-: Añadido detrás de los caracteres numéricos (#) hace que aparezca el signo (-) detrás, sólo en los valores negativos.

^^^^: Añadido detrás de los caracteres numéricos (#) permite imprimir en forma exponencial; en donde los dos primeros (^) representan el caracter (E) y el signo, y los dos últimos, las dos cifras del expònente.
PRINT USING "##.#^^^^";1000
Obtendremos: -1.0E+03

,: Si se sitúa una coma en el formato a la izquierda del punto decimal, la parte entera del número será dividida por comas cada tres dígitos.
PRINT USING "####,.#";1000.55
Se obtendrá: 1,000.6

PRINT # Instrucción BASIC (cassette y disco) que imprime datos en un archivo de tipo secuencial.
PRINT # va seguida de un número que hace referencia al canal del archivo abierto con OPEN y de una expresión numérica o alfanumérica cuyo resultado es el dato a imprimir.
Los distintos separadores cumplen diferentes misiones (Véase INPUT #).

```
PRINT #1,"ORDENADOR"
```

Imprime en el archivo abierto con el número 1 la palabra ORDENADOR.
Este archivo puede ser de cassette o disco, aunque también pueden emplearse otros dispositivos.
Esta es la forma de imprimir caracteres en el modo de gráficos 2 y 3, como se muestra en el siguiente programa:

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PSET (96,96)
40 PRINT #1,"ORDENADOR"
50 PSET (112,104)
60 PRINT #1,"M S X"
70 GOTO 70
```

PRINT # USING Instrucción BASIC (cassette y disco) que imprime datos en una archivo de tipo secuencial, con el formato determinado por una cadena máscara.

PRINT # USING lleva detrás del signo # un número que hace referencia al canal del archivo abierto con OPEN.
A continuación de USING debe situarse la cadena de máscara compuesta por caracteres especiales de formato (Véase PRINT USING).
Por último, debe situarse una expresión numérica o alfanumérica. Ejemplo:

```
PRINT #1 USING "!";"HOLA"
```

Imprime la inicial H en el archivo secuencial abierto con el número 1.

```
PRINT #1 USING "####.##";1234.567
```

Imprime el valor redondeado 1234.57 en el archivo secuencial abierto con el número 1.

prioridad de sprite Se manifiesta en el momento de solaparse dos sprites, y consiste en que el de prioridad mayor se superpone al de prioridad inferior.
Existen hasta 32 grados de prioridad o planos de sprite numerados de 0 a 31. El que lleva número inferior es el de mayor prioridad.
Solamente pueden visualizarse cuatro sprites por línea. Si apareciera un quinto sprite, desaparecería el de menor prioridad.

procesador (Véase microprocesador).

proceso de datos Hace referencia al tratamiento mediante ordenador de gran cantidad de información, especialmente en el campo del comercio y de los negocios.

proceso de interrupciones (Véase manejo de interrupciones, interrupción).

programa Serie de instrucciones pertenecientes a un lenguaje de programación, que realizan una acción concreta.
Suele ser la traducción a un lenguaje de la serie de pasos de un algoritmo.

programación del VDP (1) Desde BASIC la programación del VDP debe realizarse mediante las instrucciones: VDP, BASE, VPEEK y VPOKE.

(2) Desde código máquina se puede recurrir a las rutinas del sistema operativo que permiten realizar la mayoría de las funciones, por lo que el acceso directo no suele ser necesario.
Ambos procedimientos requieren un conocimiento en profundidad de la estructura de pantalla, del VDP y su funcionamiento.

programación estructurada Todo tipo de programación tiene una cierta estructura.
Por ejemplo, un bucle FOR ... NEXT está compuesto por una sentencia FOR de comienzo de bucle, unas instrucciones intermedias que se van a repetir y una sentencia de cierre NEXT.
Este tipo de estructuras son mucho más frecuentes en otros lenguajes de programación, en los cuales prácticamente todas las sentencias se agrupan y relacionan, hasta el punto de poderse prescindir de los números de línea.
En estos casos (PASCAL) estamos ante lenguajes estructurados.

PSET Instrucción que colorea de un color definido un punto en la pantalla de gráficos, de coordenadas (X,Y).

Las coordenadas irán entre paréntesis, pudiendo expresarse en forma absoluta o relativa (mediante STEP).
El color irá a continuación, y sèparado por una coma, siendo de aplicación los códigos de color ya conocidos.

```
PSET (128,96),15
```

Iluminará de color blanco el punto central de la pantalla.
Esta instrucción solamente puede utilizarse en los modos gráficos definidos por SCREEN 2 y SCREEN 3.

PSG Abreviaturas en inglés de Generador Programable de Sonido.

puertos de E/S Son los circuitos internos que permiten la comunicación del ordenador con el exterior.
Tienen asignados diferentes direcciones según su función, si bien estas direcciones no están incluidas en la especificación del standard MSX.
A título de ejemplo, se señalan las siguientes:

Interfaz RS-232-C:	&H80, &H81
Impresora:	&H90, &H91
VDP:	&H98, &H99
PSG:	&HA0, &HA1, &HA2
PPI:	&HA8, &HA9, &HAA, &HAB

A estas vías de acceso se pueden acceder desde BASIC mediante las instrucciones INP (lectura) y OUT (escritura), y desde código máquina con IN (lectura) y OUT (escritura).

punto flotante (Véase notación científica).

punto y coma Es un signo especial utilizado para evitar el retorno de carro automático que sigue a una sentencia PRINT.

PUSH Mnemónico de instrucción Z-80 opuesto a POP.
Sitúa el contenido de un par de registros en la pila.

PUT Instrucción de BASIC de disco empleada en el manejo de archivos de acceso directo.
Traslada al disco el valor actual de las variables de FIELD que deben haber sido escritas mediante LSET y PSET.
PUT incluye el número de registro y de archivo.
Así:

```
PUT #1,20
```

Escribe en el archivo abierto con el número 1 el registro número 20.

PUT SPRITE Instrucción BASIC que permite mostrar los sprites en pantalla.
Utiliza como parámetros:
- El número de plano de visualización, que debe estar comprendido entre 0 y 31, siendo los más bajos los de mayor prioridad.
- Las coordenadas X,Y que corresponden a la esquina superior izquierda del sprite (X comprendida entre -32 y 255, Y comprendida entre -32 y 191) en modo absoluto o relativo.
- El número de figura móvil utilizado al definirla con SPRITE$ (comprendido entre 0 y 255 para 8x8 puntos y 0 y 63 para 16x16 puntos.

Los distintos parámetros pueden ser expresiones numéricas, siempre dentro de los valores permitidos.

Si se omiten las coordenadas, el sprite aparece en el último punto referenciado.
Si se omite el número de sprite, se toma el de plano de sprite por omisión.
Si se omite el color, se adopta el de fondo.
Se producen efectos especiales con los valores de la coordenada Y: 208 y 209.
En el siguiente ejemplo se inicializa un sprite de 16x16 puntos formado por líneas horizontales, que se ha definido mediante la función STRING$. Este sprite se mueve según una función sinusoidal asignada a la variable A, que aparece como ordenada de la instrucción PUT SPRITE. El color de visualización es azul celeste. La trayectoria de las esquinas del sprite se traza mediante instrucciones PSET en el color por omisión (blanco).

```
 10 COLOR 15,4,4
 20 SCREEN 2,2
 30 DEFINT X,A
 40 SPRITE$(0)=STRING$(32,85)
 50 CLS
 60 FOR A=0 TO 239
 70 A=88+80*SIN(X/38)
 80 PUT SPRITE(0),(X,A),7,0
 90 PSET (X+1,A+1)
100 PSET (X+1,A+16)
110 PSET (X+15,A+16)
120 PSET (X+15,A+1)
130 NEXT X
140 GOTO 50
```


QWERTY Se dice de la disposición habitual de las teclas de teclado (anglosajona), opuesta a la disposión AZERTY (francesa).

radian Unidad angular eqivalente a una semicircunferencia, 180 grados sexagesimales o 200 grados centesimales.

raiz cuadrada La raiz cuadrada de un argumento es aquel número que multiplicado por sí mismo dá como resultado el mismo argumento.
En el ordenador se representa por SQR.
No pueden utilizarse argumentos negativos.

RAM Abreviaturas en inglés de Memoria de Acceso Aleatorio.
(Véase memoria).

ranura Cada uno de los cuatro bloques (ranura primaria) de 64K de memoria de donde pueden proceder las 4 páginas de 16K controladas por el procesador.
Pueden dividirse, a su vez, en otras cuatro ranuras secundarias.

raqueta Mando de juego simplificado que dispone de movimiento en una única dirección.

READ Instrucción BASIC que asigna a una lista de variables los sucesivos valores contenidos en una instrucción DATA.
La instrucción READ va seguida por una serie de variables separadas por comas.
Cuando el programa alcanza la primera instrucción READ, busca la sentencia DATA con número de línea más bajo, y asigna el primer valor de DATA a la primera variable de READ, y así, sucesivamente.
La siguiente instrucción READ asignará a su primera variable el valor de DATA siguiente al último leido.

Para alterar este proceso secuencial de asignación se utiliza la instrucción RESTORE o RESTORE número de línea. La primera hace que el siguiente READ vuelva a comenzar a asignar desde la sentencia DATA de número más bajo.
La segunda obliga a READ a leer desde la DATA señalada por el número de línea o inmediatamente si no se especifica éste.
Los tipos de variables y valores a asignar deben ser iguales. En caso contrario, se produce un error "Type mismatch".
No se deben leer más valores que los contenidos en DATA para evitar un error "Out of DATA".
READ, DATA y RESTORE forman una estrctura de programación utilizada para asignar valores a variables desde el propio programa.
El siguiente ejemplo se emplea para almacenar una pequeña rutina de máquina.

```
10 CLEAR 200,50000
20 DC=50001
30 FOR X=0 TO 16
40 READ A
50 POKE DC+X,A
60 NEXT X
70 DEF USR=50001
80 R=USR(0)
90  DATA  205,108,0,1,83,3,251,65,205,162,0,
11,120,177,32,246,201
```

recursión Es un proceso utilizado en programación por el cual una subrutina llama a otra y ésta a su vez a otra, y así, sucesivamente.

recursiva Se dice de las subrutinas que son llamadas o utilizadas por otras.

red local Sistema que permite la interconexión de varios ordenadores entre sí, de forma que puedan compartir datos y dispositivos como impresoras, discos rígidos, etc.
Es muy utilizada en aulas informáticas.

redondeo Convenio seguido para al eliminación de cifras decimales, por el cual, si una cifra decimal es 5 o superior a 5, la siguiente cifra más significativa aumenta una unidad, y si es inferior a 5, queda como está.

3.5 redondeado es 4
3.49 redondeado es 3

registro Son pequeñas unidades de memoria alojadas en el microprocesador que permiten el intercambio de datos entre la unidad de control, la unidad aritmético-lógica y la memoria.
En el procesador Z-80 se destacan los siguientes:

A	Acumulador
F	Banderas
BC	Registros de utilización genérica.
DE	
HL	
I	Interrupción
R	Refresco
IX	Indice X
IY	Indice Y
SP	Puntero de pila
PC	Contador de programa

Además de estos, existe un juego alternativo de registros compuesto por:

A'	Acumulador
F'	Banderas
B'C'	Registros de utilización genérica
D'E'	
H'L'	

relación Es un tipo especial de operación. (Véase operación de relación).

reloj interno El procesador precisa para su funcionamiento de un reloj, que mediante sus impulsos, señale la ejecución de las distintas instrucciones, acceso a memoria, etc.
Es un reloj de tipo electrónico (oscilador) controlado por un cristal de cuarzo funcionando a 3.58 MHz.

REM Se utiliza para escribir comentarios dentro de un programa para su aclaración. Puede sustituirse por un apóstrofe.
Cualquier instrucción colocada detrás de REM en la misma línea, no es ejecutada.
Ejemplo:

```
10 REM PROGRAMA DE CALCULO
......
100 REM ENTRADA DE DATOS
......
200 REM CALCULO
......
300 REM SALIDA DE RESULTADOS
......
```

RENAME Instrucción del sistema operativo de disco (MSX-DOS) que permite modificar el nombre de un archivo.

```
RENAME A: PROG1.BAS PROG2.BAS
```

Cambia el nombre del programa BASIC PROG1 por PROG2.

RENUM Instrucción que numera de nuevo las líneas de un programa.

Esta instrucción respeta las direcciones que toman las instrucciones GOTO, GOSUB, ON ... GOTO, ON ... GOSUB, THEN, ELSE, IF THEN y ERL, modificando sus números asignados.
Si falta el número de línea al que tiene que saltar alguna de estas instrucciones, aunque se ejecute RENUM permanece este error, por falta de número de línea de salto.
Esta instrucción tiene tres parámetros, todos se pueden omitir y van separados por comas, estos son:

- Número de línea inicial, el nuevo número a partir del cual queremos que numere el programa, en caso de omisión toma el valor 10.
- El número antiguo a partir del cual queremos renumerar el programa, en caso de omisión empezará por la primera.
- Incremento deseado para cada línea, será constante para todas y por omisión toma el valor 10.

Utilizada sin parámetros, renumera las líneas empezando por 10 y con incrementos de 10.

representación binaria (Véase notación binaria).

RES Mnemónico de instrucción Z-80 que opera sobre bits aislados pertenecientes a direcciones de memoria, o registros, poniéndolos a cero.

resolución Característica de los distintos modos gráficos de pantalla, que se mide por el número máximo de puntos que pueden visualizarse en pantalla.
En general, se habla de alta resolución cuando existen más de 40000 puntos.

En nuestro caso, el modo dado por SCREEN 2 es de alta resolución, y el de SCREEN 3 es de baja resolución.
También permite medir la calidad de un monitor.

RESTORE Instrucción BASIC, utilizada para comenzar el proceso READ-DATA desde la primera constante del DATA señalado con un número de línea más bajo.
Esta instrucción permite la introducción de un parámetro, correspondiente al número de línea por la cual se comienza la lectura de las variables de otro DATA. Si en ese número de línea no hay un DATA que asigne unos valores a las variables de READ, entonces el ordenador buscará una sentencia DATA en las siguientes líneas.
(Véase READ, DATA).

RESUME Una instrucción BASIC que finaliza la ejecución de una rutina de tratamiento de errores, a la que se bifurca el programa mediante ON ERROR GOTO.
Puede realizar tres acciones diferentes:
- Volver a la sentencia donde se ha producido el error.
  Para ello se utiliza RESUME ó RESUME 0.
- Volver a una sentencia determinada.
  Se emplea RESUME número de línea.
- Volver a la sentencia siguiente a la del error.
  Se emplea RESUME NEXT.
  (Véase ON ERROR GOTO).

RET Mnemónico de instrucción Z-80 que ejecuta el retorno de una subrutina a la que se ha llamado desde BASIC con USR, o desde código máquina con CALL.

El retorno puede ser condicional, en función de los valores del registro de banderas.

RETI Mnemónico de instrucción Z-80 semejante a RET, que elimina la petición de interrupción.

RETN Mnemónico de instrucción Z-80 semejante a RETI, pero utilizada para interrupciones no enmascarables.

RGB Abreviaturas en ingles de rojo, verde y azul.
Designa un tipo especial de modulación de señal para monitores, por el cual las tres señales de color se envían separadamente.
Produce una calidad superior a la modulación de video compuesto o TV.

RIGHT$ Función alfanumérica de BASIC utilizada para actuar sobre cadenas alfanuméricas, que devuelve X caracteres de esa cadena, contados desde la derecha.
Si X excede el número de caracteres de esa cadena, entonces se obtendrá solamente todos los caracteres de dicha cadena. Si X es nulo, se obtendrá una cadena nula. Si X es decimal, se ignorarán los números situados detrás del punto.
Si tenemos en cuenta que algunos caracteres están compuestos por códigos de 2 bytes (siendo el primero CHR$(1)), cuando se aplica la función RIGHT$ a una cadena definida con estos caracteres, el número de ellas que se visualizarán será la mitad de los indicados, puesto que cada uno de ellos se considerará formado por dos caracteres.

```
10 FOR N=64 TO 96
20 A$=A$+CHR$(1)+CHR$(N)
30 NEXT N
40 PRINT A$
50 PRINT RIGHT$(A$,16)
```

Este ejemplo crea una cadena a base de caracteres de 2 códigos. Al ejecutar la línea 50, sólo visualizará 8 caracteres gráficos en pantalla, puesto que éstos están considerados en la línea 20, como formados por dos caracteres.

RL Mnemónico de instrucción Z-80 que desplaza hacia la izquierda una posición el contenido de un registro o una dirección de memoria.
El contenido de la bandera de exceso se sitúa en el bit menos significativo, y el bit más significativo se traslada a la bandera de exceso.

RLA Mnemónico de instrucción Z-80 semejante a RL pero para el registro A.

RLC Mnemónico de instrucción Z-80 semejante a RL, pero donde el contenido del bit más significativo se traslada a la bandera de exceso y al bit menos significativo.

RLCA Mnemónico de instrucción Z-80 semejante a RLC pero para el registro A únicamente.

RLD Mnemónico de instrucción Z-80 que realiza una rotación o traslado entre tres números en notación BCD.

RND Función que obtiene un número aleatorio menor que 1 y mayor o igual a cero.

Este valor aleatorio se obtiene a partir de un generador seudoaleatorio que contiene una serie ya prefijada de números, y según sea el parámetro que le demos funcionará de modo diferente.
Si X es un número positivo mayor que cero, se obtiene el siguiente número en la serie. Si es igual a cero, se obtiene el mismo número que la vez anterior, y si es negativo, se inicializa la serie en algún lugar desconocido.
La forma de obtener una serie lo más aleatoria posible es usando como parámetro X=-TIME ya que se inicializará siempre la serie y en lugares diferentes.

ROM Abreviaturas en ingles de Memoria de Sólo Lectura.
(Véase memoria).

RR Mnemónico de instrucción Z-80 que desplaza hacia la derecha una posición el contenido de un registro o una dirección de memoria.
El contenido de la bandera de exceso se sitúa en el bit más significativo y el bit menos significativo se traslada a la bandera de exceso.

RRA Mnemónico de instrucción Z-80 semejante a RR, pero para el registro A.

RRC Mnemónico de instrucción Z-80 semejante a RR, pero donde el contenido del bit menos significativo se traslada a la bandera de exceso y al bit más significativo.

RRCA Mnemónico de instrucción Z-80 semejante a RRC, pero para el registro A únicamente.

RRD Mnemónico de instrucción Z-80 que realiza una rotación o traslado entre tres números en notación BCD.

RSET Instrucción de BASIC de disco utilizada en el tratamiento de archivos de acceso directo semejante a LSET, pero justificando por la derecha.

RST Mnemónico de instrucción Z-80 . Realiza un salto a una de las ocho direcciones siguientes:

&H00, &H08, &H10, &H18, &H20, &H28, &H30 y &H38.

Se puede utilizar como salida de una interrupción.

RS-232-C Protocolo de comunicaciones de serie, con un único conductor a través del que se envían las señales. La transmisión de un byte requiere de hasta once señales.

RUN (1) Instrucción BASIC que ejecuta el programa almacenado en la memoria. Si se omite el número de línea, el programa es ejecutado desde la primera línea.

RUN 100 ejecuta el programa en memoria, a partir de la sentencia 100.

(2) Instrucción BASIC que permite cargar un programa en ASCII desde cassette y ejecutarlo, siendo semejante a:

```
LOAD "nombre de programa",R
```

Ejemplo:

```
RUN "CAS:PROG1"
```

Carga y ejecuta el programa PROG1.

rutina (Véase subrutina).

salida de impresora La salida de impresora del sistema MSX es del tipo Paralelo-Centronics.
Existen impresoras específicas para imprimir todos los caracteres del sistema.
Para este tipo de impresoras debe especificarse:

```
SCREEN ,,,,0
```

Para impresoras que no sean del sistema, podrá especificarse:

```
SCREEN,,,,1
```

o cualquier valor distinto de cero.
En este caso, los caracteres gráficos se envían como espacios.

salto Es una modificación en el flujo normal del programa.
Se pueden realizar incondicionalmente mediante GOTO o en base a una condición con IF ... THEN o ON ... GOTO.

SAVE Instrucción que almacena un archivo de programa BASIC en el dispositivo indicado en código ASCII.
La ventaja del código ASCII es que nos permite fusionar un programa en memoria con el que esté almacenado en este código mediante una sentencia MERGE.
Los nombres de dispositivos son los usuales que allí comentamos.
Ejemplo:

```
SAVE"CAS:PROG1"
```

SBC Mnemónico de instrucción Z-80 que resta de A ó de HL un valor dado por una constante, registro o posición de memoria más el valor de la bandera C de exceso.

SCF Mnemónico de instrucción Z-80 que asigna el valor 1 a la bandera C.

SCREEN Instrucción para seleccionar el modo de pantalla deseado, además del tamaño de sprite, si se desea o no que se produzca el eco del teclado, la velocidad de transferencia del cassette y el tipo de impresora.

1er.parámetro	Modos de Pantalla
0	Texto con 40 columnas x 24 líneas
1	Texto con 32 columnas x 24 líneas
2	Gráficos alta definición
3	Gráficos multicolor

2º parámetro	Tamaño de Sprite
0	8 x 8 puntos sin ampliar
1	8 x 8 puntos ampliados
2	16x16 puntos sin ampliar
3	16x16 puntos ampliados

3er.parámetro	Eco de Teclado
0	No
≠0	Si

4º parámetro	Velocidad de Transferencia
1	1200 baudios
2	2400 baudios

5º parámetro	Impresora
0	Impresora MSX
≠0	Impresora no MSX

Los símbolos gráficos aparecerán en forma de espacio. (Impresora no MSX).
En caso de omisión de estos parámetros, se quedan en el estado en el que estaban.

Los valores de inicialización son:

0	Modo de pantalla
0	Tamaño de sprite
1	Eco de teclado
1	Velocidad de transmisión
0	Impresora MSX

sentencia Es una instrucción completa con sus parámetros.
Si es ejecutada en modo directo, es una orden o comando.
Si es ejecutada en modo indirecto, deberá ir precedida por un número de línea.

serie Uno de los protocolos de transmisión de señales.
Utiliza un único conductor por donde se van enviando sucesivamente las señales.
(Véase RS-232-C).

SET Mnemónico de instrucción Z-80 que opera sobre bits aislados pertenecientes a direcciones de memoria o registros, asignándoles el valor 1.

seudoaleatorio (Véase aleatorio, RND).

SGN La instrucción SGN(X) ofrece el valor 1 si X es mayor que cero, cero si X es igual que cero, y -1 si X es menor que cero.
El siguiente programa simula el funcionamiento de INT mediante SGN y FIX.

```
100 INPUT X
110 P=FIX(SGN(X)*(FIX(ABS(X))+1+(SGN(X)*
    (-.5))
120 PRINT" ";INT(X)
130 PRINT" ";P
```

SIN La instrucción SIN(X) calcula, en radianes, el valor del seno de X.

```
PRINT SIN(4*ATN(1)/2)
```

Nos dá el valor 1 correspondiente a PI/2.

sintetizador de voz Es un dispositivo que lleva programadas una serie de secuencias sonoras que imitan la voz humana (fonemas).
Su combinación permite imitar la pronunciación y dicción humanas.

sistema operativo Son un conjunto de rutinas que controlan el funcionamiento del ordenador y la gestión de los distintos dispositivos conectados al mismo.
En el MSX-BASIC estas rutinas se encuentran juntamente con el intérprete, en la memoria ROM.

sistema operativo de disco Las rutinas residentes en ROM del sistema operativo del MSX-BASIC no son adecuadas para la gestión de discos.
Para ello se han de cargar otras rutinas que constituyen el sistema operativo de disco (Disk Operative System DOS).
El sistema operativo standard es el MSX-DOS, si bien también puede utilizarse el CP/M 2.2.
En algunos casos, (RAM menor de 64K) estas rutinas se cargan de un cartucho.

SLA Mnemónico de instrucción Z-80 que desplaza el contenido de un registro o posición de memoria hacia la izquierda, cargando el valor 0 en el bit menos significativo y modificando, si procede, la bandera C.

software Denominación genérica por la que se conocen los programas que se utilizan en el ordenador.
Se distingue del hardware que es la "maquinaria" (circuitos integrados, conexiones, componentes) del ordenador y del firmware que constituyen los programas incorporados dentro del ordenador (el intérprete BASIC y el sistema operativo en el MSX-BASIC).

sonido El sistema MSX dispone de buenas posibilidades para la generación de sonido (tres canales, ocho octavas, control de envolvente, generación de ruido, etc.).
(Véase Generador Programable de Sonido, PLAY , SOUND).

SOUND Instrucción BASIC que permite la generación de efectos de sonido mediante la escritura directa de los registros del PSG.
Esta instrucción va seguida del número de registro y separado por una coma, del valor numérico que deseamos asignar al registro.
El PSG dispone de 13 registros, accesibles con SOUND, cuyas funciones se relacionan a continuación:

Registro 0: Es el ajuste fino de frecuencia de tono del canal A. Su valor debe estar comprendido entre 0 y 255.

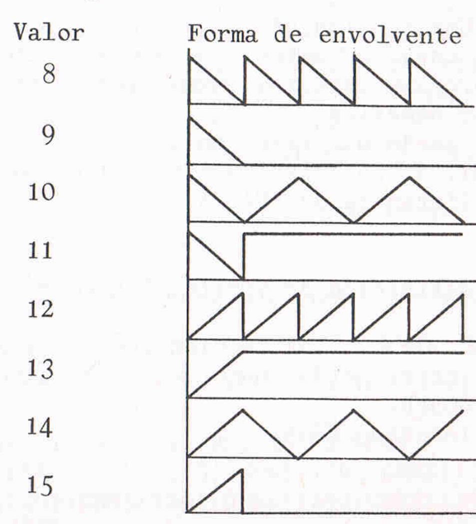
Registro 1: Es el ajuste grueso de frecuencia de tono del canal A. Su valor debe estar comprendido entre 0 y 15.
La frecuencia de tono en Hz. es aproximadamente igual a:
1996750/16*((Registro1x256)+Regist.0)

Registro 2: Semejante al Registro 0, pero para el canal B.
Registro 3: Semejante al registro 1, pero para el canal B.
Registro 4: Semejante al registro 0, pero para el canal C.
Registro 5: Semajante al registro 1, pero para el canal C.
Registro 6: Es el ajuste de frecuencia de ruido.
La frecuencia de ruido en Hz. es aproximadamente igual a:
1996750/16*Registro 6
Registro 7: Este registro es el que conecta o desconecta los tres canales de sonido para tono o ruido.
Cada canal se controla con un bit del registro, siendo 0 activado y 1 desactivado, según la distribución siguiente:

	ruido			tono		
bit	5	4	3	2	1	0
canal	C	B	A	C	B	A

Los bits 7 y 6 deben ser siempre 1.
Por lo tanto, para conectar el canal C en ruido, y el B y A en tono, formamos el valor binario:
11011100
cuyo equivalente decimal es 220.
Registro 8: Permite controlar la amplitud (volumen) del canal A, y en su caso, activar el control de envolvente.
El volumen se regula con valores comprendidos entre 0 y 15. Si se utiliza el valor 16, el volumen queda bajo el control de la envolvente (registros 11, 12 y 13).

Registro 9: Semejante al registro 8, pero para el canal B.
Registro 10: Semejante al registro 8, pero para el canal C.
Registro 11: Es el ajuste fino de frecuencia de envolvente. Su valor debe estar comprendido entre 0 y 255.
Registro 12: Es el ajuste grueso de frecuencia de envolvente. Su valor debe estar comprendido entre 0 y 255.
La frecuencia de tono en Hz. es aproximadamente igual a:
1996750/256*((Registro12*256)+ Registro 11)
Registro 13: Determina la forma de envolvente.
Existen 8 formas a las que corresponden los siguientes valores del registro 13:



A continuación, se muestran algunos ejemplos de la utilización de SOUND:

```
10 SOUND 7,247
20 SOUND 8,16
30 SOUND 12,100
40 SOUND 13,14
```

Este programa activa el canal A para ruido con control de envolvente número 14.
Con la frecuencia determinada por el valor 100, se obtiene una cadencia semejante a una ola.

SPACE$ Una función BASIC que genera una cadena formada por el caracter espacio. El número de espacios estará definido como un valor numérico.

```
PRINT "ORDENADOR"; SPACE$(10);"MSX"
```

Separa con diez espacios las palabras ORDENADOR MSX.

SPC Una función BASIC que genera un cierto número de espacios entre dos mensajes de pantalla. El número de espacios podrá venir definido como un valor numérico.
Sólo se puede utilizar con las instrucciones PRINT y LPRINT, al no constituir por sí misma una cadena, a diferencia de SPACE$.

sprite
(Véase definición de sprite, figura móvil).

SPRITE OFF Instrucción que cancela el proceso de interrupción declarado previamente por ON SPRITE GOSUB.
(Véase interrupción).

SPRITE ON Instrucción que dá validez al proceso de interrupción por choque de sprites declarado previamente por ON SPRITE GOSUB.
(Véase interrupción).

SPRITE STOP Instrucción que suspende y memoriza el proceso de interrupción declarado previamente por ON SPRITE GOSUB hasta encontrar una sentencia SPRITE ON o RETURN.

SPRITE$ Es una variable especial de BASIC que permite la definición de sprites.
Para su utilización se debe especificar el número de sprite entre paréntesis, que debe estar comprendido entre 0 y 255 para sprites de 8x8 puntos, y entre 0 y 63 para 16x16 puntos.
A la variable así obtenida se le debe asignar una cadena formada por 8 ó 32 caracteres (para sprites de 8x8 ó 16x16 puntos, respectivamente), cada uno de estos caracteres representa una línea de 8 puntos del sprite, y su código ASCII debe coincidir con el equivalente decimal del valor binario, del conjunto de puntos que forma la línea.
Supongamos que queremos hacer un sprite con forma de cuadrado de 8x8 puntos.
La primera línea estará totalmente llena, y por lo tanto, su valor binario será:

11111111 = 255

La línea siguiente tendrá un punto al principio y un punto al final:

10000001 = 129

y así sucesivamente hasta llegar a la línea final de valor 255.
Para asignar el valor a SPRITE$ hacemos:

```
SPRITE$(0)=CHR$(255)+CHR$(129)+CHR$(129)+
CHR$(129)+CHR$(129)+CHR$(129)+CHR$(129)+
CHR$(255)
```

(Véase definición de sprite).

SQR Calcula, en forma numérica, la raiz cuadrada de una expresión numérica.
La expresión debe ser positiva para evitar un error del tipo (5) Illegal function call.

SRA Mnemónico de instrucción Z-80 que desplaza hacia la derecha una posición el contenido de un registro o posición de memoria, a excepción de su bit más significativo, que permanece invariable.
El segundo bit más significativo adquiere el valor cero y el contenido del bit menos significativo se traslada a la bandera de exceso C.

SRL Mnemónico de instrucción Z-80 que desplaza hacia la derecha una posición el contenido de un registro o posición de memoria.
El bit más significativo adquiere el valor cero y el menos significativo se transfiere a la bandera de exceso.

STEP Instrucción utilizada para cambiar el incremento del contador de un bucle FOR ... NEXT que vale 1 por omisión.
Los valores numéricos que definen al STEP, pueden ser tanto positivos como negativos.
En el caso de un STEP positivo, el incremento del contador variará hacia un valor mayor; en un STEP negativo, variará hacia un valor menor.
(Véase FOR ... NEXT).

STICK Función BASIC utilizada para la lectura de la posición de las teclas del cursor y de los mandos de juegos.

El argumento de la función STICK será:

- 0, si el movimiento se controla mediante las teclas del cursor.
- 1, si el movimiento se controla mediante el mando de juegos conectado en A.
- 2, si el movimiento se controla mediante el mando de juegos conectado en B.

La utilización de la función STICK es acumulativa, es decir, que en un programa pueden utilizarse los mandos de juegos A y B, y las teclas del cursor simultaneamente.
El argumento debe ir entre paréntesis.
El siguiente ejemplo utiliza un mando de juegos para desplazar un caracter # por la pantalla. Para usarlo con las teclas de cursor, debe sustituirse el parámetro 1 de STICK por 0.

```
1010 CLS
1020 N=15
1030 LOCATE 10,N:PRINT " ";
1040 J=STICK(1)
1050 IF J=0 THEN LOCATE 10,N:PRINT"#"
1060 IF J=1 THEN N=N+1:IF N > 20 THEN N=20
1070 IF J=5 THEN N=N-1:IF N < 0 THEN X=0
1080 LOCATE 10,N:PRINT"#"
1090 GOTO 1030
```

STOP Instrucción utilizada para interrumpir la ejecución del programa.
Para la reanudación de la ejecución, bastará con utilizar la instrucción CONT.
Si después de la interrupción alteramos alguna instrucción, el ordenador no podrá proseguir con CONT, deberá utilizarse GOTO número de línea.

STOP OFF Instrucción que cancela el proceso de interrupción declarado previamente por ON STOP GOSUB.
(Véase interrupción).

STOP ON Instrucción que dá validez al proceso de interupción generado al pulsar las teclas CTRL + STOP declarado previamente por ON STOP GOSUB.
(Véase interrupción).

STOP STOP Instrucción que suspende y memoriza el proceso de interrupción declarado previamente por ON STOP GOSUB hasta encontrar una sentencia STOP ON ó RETURN.

STRIG Función BASIC para la lectura de los disparadores de los mandos de juegos o de la barra de espacio.
Retorna el valor -1 si se encuentran pulsadas y cero en caso contrario.
Utiliza como argumentos los valores siguientes:

Barra de espacio	0
Disparador mando A	1,3
Disparador mando B	2,4

Ejemplo:

```
PRINT STRIG (0)
```

Imprime el valor -1 si estamos pulsando la barra de espacio al pulsar RETURN.

STRIG OFF Instrucción que cancela el proceso de interrupción declarado previamente por ON STRIG GOSUB.
(Véase interrupción).

STRIG ON Instrucción que dá validez al proceso de interrupción generado al pulsar la barra espaciadora o el disparador de un mando de juegos, declarado previamente por ON STRIG GOSUB.
(Véase interrupción).

STRIG STOP Instrucción que suspende y memoriza el proceso de interrupción declarado previamente por ON STRIG GOSUB hasta encontrar una sentencia STRIG ON ó RETURN.

STR$ Función BASIC que transforma un valor numérico en otro alfanumérico. Es la instrucción opuesta a VAL.
Permite asignar un valor numérico a una cadena.

```
PRINT LEN (STR$(100))
```

Imprime el valor 4, pues se considera que todos los valores numéricos llevan un signo, sea éste positivo o negativo.

STRING$ Genera una cadena con un número especificado de caracteres de un tipo determinado.
Utiliza como argumentos el número de caracteres a repetir, y el código del que debe repetirse.
Así:

```
PRINT STRING$ (200,65)
```

Imprime una cadena formada por la letra A repetida doscientas veces.

SUB Mnemónico de instrucción Z-80 que resta un número, el contenido de un registro o de la memoria del acumulador, sin afectar a la bandera de exceso.

subcadena Es una parte de una cadena superior. Se obtiene como resultado de la aplicación de instrucciones como LEFT$, MID$ o RIGHT$.

subíndice Un valor numérico que determina un elemento de una variable de conjunto unidimensional o la fila o columna de una multidimensional

DIM A(10,20)

Los valores 10 y 20 expresan los máximos subíndices utilizables.

A(5,15)

Expresa el elemento de la fila 5 (6 si contamos la 0), columna 15 (16 si contamos la 0).

subrutina Una subrutina es un fragmento de programa que realiza una acción concreta.
Si esta acción precisa realizarse repetidamente, en vez de repetirla cada vez, se puede dejar aparte del programa principal y llamarla cuando se precise mediante GOSUB.
A tal fin, la subrutina deberá terminar con la instrucción RETURN y estará separada del programa mediante STOP, END o será saltada con GOTO.
Si una subrutina llama a otra estaremos ante una subrutina recursiva.
(Véase GOSUB, RETURN).

SWAP Intercambia los contenidos de dos variables del mismo tipo, independientemente de que tengan mayor o menor valor numérico o alfanumérico.

SYSTEM Una instrucción de sistema operativo de disco utilizada para retornar, desde BASIC, al ámbito del sistema operativo. Al tenerse que ejecutar desde BASIC, debe ir precedida por CALL ó _ (subrayado).

TAB Instrucción utilizada en combinación con PRINT o LPRINT, para situar un mensaje en pantalla o en la impresora, a un número especificado de posiciones de carácter, comenzando desde el margen izquierdo.

tabla de verdad Expresa todas las posibles combinaciones de operandos lógicos en una operación lógica y sus respectivos resultados.
(Véase operación lógica).

tabla de atributos de sprite Almacena las coordenadas, el color y el número de patrón de cada sprite. Comienza en la dirección 6912 de la VRAM.
La posición 6912 guarda la coordenada Y del sprite cero, la 6913 almacena la coordenada X, la 6914 el número de patrón y la 6915 el bit previo de reloj en su bit más significativo y el color en los cuatro menos significativos (el resto no se usan).
Por lo tanto, esta tabla tiene 32 x 4 = 128 posiciones, de la 6912 a la 7039.
(Véase bit previo de reloj).

tabla de color Almacena los colores de visualización de cada elemento de la tabla de nombres de patrones.
Existe, solamente, en los modos de pantalla 1 y 2. Comienza en la dirección 8192.
En el modo 1 consta de 32 direcciones, cada una de las cuales afecta a 8 elementos de la tabla de patrones generadores.

Cada byte representa en sus cuatro bits más significativos el color de texto y en los cuatro siguientes el color de fondo.
En el modo 2, cada dirección define el color de una dirección de la tabla de patrones.
Por lo tanto, en cada segmento horizontal coincidente con una posición de carácter, solamente pueden existir dos colores.
Los colores se definen igual que en el modo 1.
El número total de elementos de la tabla de color en modo 2 es de:

256 x 192 /8 = 6144 bytes

tabla de nombres de patrones Almacena el nombre de patrón que va a visualizarse en cada posición de carácter.
Existe en todos los modos de pantalla, siendo su dirección de comienzo:

Modo 0	0
Modo 1	6144
Modo 2	6144
Modo 3	2048

El modo 0 consta de 960 elementos (24 filas x 40 columnas). Cada uno representa el código ASCII del carácter que se va a visualizar en su posición respectiva.
El modo 1 está formada por 768 elementos (24 filas x 32 columnas) y actúa en forma semejante al modo 0.
El modo 2 precisa de 768 elementos (los mismos que en el modo 1), si bien en vez de representar el código ASCII del carácter representan el número de un elemento de tabla de patrones generadores.
Como cada byte sólo puede seleccionar un valor entre 0 y 256 y la tabla de patrones dispone de 768 elementos, ambas tablas se dividen en tres

zonas de forma que cada elemento de una zona de la tabla de nombres llama a elementos de su zona respectiva de la tabla de patrones.
En el modo 3, la tabla de nombres también consta de 768 elementos. Sin embargo, la tabla de patrones contiene los colores de los bloques de 4x4 puntos característicos de este modo.
Para cada posición de carácter, se precisan dos bytes, por lo que la tabla de patrones consta de:

24 x 32 x 2 = 1536 bytes

dividido en 192 grupos de 8.
En función de la línea de carácter, se selecciona el par de bytes de cada grupo de 8.
La línea superior selecciona los dos primeros, la segunda los dos siguientes, etc.

tabla de patrones generadores Almacena los patrones o formas de puntos que componen los caracteres o formas visualizadas en pantalla.
Se utiliza en todos los modos, siendo su dirección de comienzo la 2048 en el modo 0, y 0 en los tres restantes modos.
En el modo 0 esta tabla consta de 2048 direcciones agrupadas en bloques de 8, cada uno de los cuales contiene la forma de uno de los caracteres copiados de la ROM.
En el modo 1 es semejante a la del modo 0, pero con comienzo en la dirección 0.
En el modo 2 la tabla de patrones tiene 6144 direcciones, es decir, un bit por punto. Se divide al igual que los modos 0 y 1, en bloques de 8 bytes, y además, en tres zonas de 2048 bytes (ó 256 bloques de 8).
Cada bloque de 8 bytes de una de las zonas es seleccionado mediante un byte de la tabla de nombres correspondientes a su zona.

En el modo 3 la tabla de patrones contiene los colores de los bloques de 4x4 puntos característicos de este modo.
Cada color de un bloque se determina con medio byte (1 nibble), por lo que una posición de carácter (4 bloques de 4x4 puntos) se representa con 2 bytes.
Por lo tanto, la tabla de patrones consta de:

24 x 32 x 2 = 1536 bytes

divididos en 192 grupos de 8.
En función de la línea de caracter se selecciona el par de bytes de cada grupo de 8.
La línea superior selecciona los dos primeros, la segunda los dos siguientes, y así sucesivamente.

tabla de patrones de sprite Almacena la forma o patrón de puntos que compone un sprite.
Se utiliza en todos los modos, a excepción del de texto de 40 columnas.
Su dirección de comienzo es 14336.
Si se utilizan patrones de 8x8, se podrán almacenar 256 patrones de ocho bytes cada uno (2048 bytes en total).
Si se emplean patrones de 16x16, se podrán almacenar hasta 64 patrones.

tablero digitalizador Es un dispositivo que permite la introducción de coordenadas manualmente, señalándolas con un puntero, sobre la superficie del tablero.
La función utilizada en su lectura es PAD (argumento).

TAN Calcula el valor de la tangente de un ángulo dado en radianes.

teclado Es el dispositivo de entrada por excelencia.
Su lectura se realiza mediante las rutinas del sistema operativo.
(Véase exploración del teclado).
La mayoría de los aparatos comercializados bajo el standard MSX utilizan teclados del tipo QWERTY, si bien en Francia se dispone de teclados AZERTY.
En los modelos originales japoneses existía una opción para caracteres Kanji.
El teclado dispone de unas 48 teclas alfanuméricas, 12 teclas generales, 5 teclas de función, 4 de edición y 4 teclas de control del cursor.

texto multicolor Aunque normalmente el texto se visualiza en dos colores, es posible crear texto en diferentes colores, en el modo 1.
Para ello debe alterarse la tabla de color.
(Véase programa ejemplo de atributo).

TIME Una variable especial de BASIC utilizada para conocer el valor del reloj interno del sistema.
Su valor inicial es cero, en el momento de encender el ordenador, y va incrementándose alrededor de 50 veces por segundo.
El valor de TIME se puede reinicializar en cualquier momento, tecleando TIME=0.
Asímismo, puede ser empleado en combinación con la función RND, con signo negativo, de tal manera que el número aleatorio generado por RND sea aún más aleatorio.

A = RND(-TIME)

Permite generar tiempos.

transferencia de bloques Es un procedimiento utilizado en código máquina para trasladar bloques de memoria de una a otra posición. Es muy útil para la animación de imágenes, debido a su velocidad.
Existen numerosas instrucciones asociadas a este procedimiento, como LDIR, LDDR, OTDR, OTIR ...

TRON Instrucción por la cual el ordenador visualiza en la pantalla el número de la línea que va ejecutando, a la vez que ejecuta dicho programa.
Si el modo de pantalla es modificado y se encuentra en el modo de gráficos, no se visualiza en la pantalla el número de cada línea.
Generalmente se utiliza para la corrección o modificación de un programa. Se puede utilizar tanto en modo directo, como indirecto.

TROFF Instrucción utilizada para desactivar la actividad de la instrucción TRON, es decir, elimina la visualización en pantalla del número de línea que va ejecutando.
Al igual que TRON, TROFF también puede ser utilizado, tanto en modo directo, como indirecto.

truncado Truncar un valor numérico consiste en obtener su parte entera.

TYPE Una instrucción del sistema operativo de disco que permite visualizar el contenido de los archivos.
Para ello debe ir seguida del nombre del archivo.

```
TYPE A: PROG1.BAS
```

Visualizará el listado del programa PROG 1.
Para que sea legible, éste debe estar almacenado en formato ASCII.

unidad central de proceso Es el centro del ordenador, entre sus funciones están:

- Llevar el control de todo el sistema según el sistema operativo que ha recibido.
- Realizar las operaciones lógicas y aritméticas que le son ordenadas.
- Ocuparse de las comunicaciones con los distintos dispositivos.

Para ello se sirve de las siguientes elementos:

- Unidad de control.
- Unidad aritmético-lógico.
- Memoria principal.
- Buses.
- Registros.

Lo que normalmente se conoce como microprocesador, suele contener la unidad de control, la unidad aritmético-lógica, las salidas para los buses y los registros.

USR Una función BASIC que cede el control del ordenador a una rutina en código máquina, con comienzo en la dirección especificada mediante:

```
DEFUSR
```

Una vez ejecutada la rutina, el resultado obtenido (en muchos casos sin sentido ni posible utilización) es entregado por la función USR.
Asímismo, USR permite el paso de parámetros a la rutina en código máquina desde BASIC.
Si hacemos:

```
A = USR 1(X)
```

Se cede el control a la rutina situada a partir de la dirección especificada por DEFUSR 1.
Una vez ejecutada, el valor obtenido se asigna a A.
El parámetro se pasa mediante X, una variable, cuyo tipo determina en qué direcciones de memoria

se va a almacenar dicho parámetro.
Si la variable es del tipo entero, su valor se almacena en las posiciones &HF7F8 y &HF7F9.
Si es de precisión sencilla, las posiciones de almacenamiento son desde la &HF7F6 a la &HF7F9.
Si es de precisión doble, las posiciones van desde la &HF7F6 a la &HF7F9.
En el caso de las variables de cadena, las posiciones &HF7F8 y &HF7F9 contienen la dirección del descriptor de cadena, que está compuesto por tres bytes, el primero es la longitud de la cadena y los dos siguientes contienen la dirección donde se encuentra.

VAL Función BASIC que permite la obtención de datos numéricos a partir de datos alfanuméricos.
Si en la variable alfanumérica del argumento de una función VAL hay más de un dato, sólo se obtendrá el dato numérico correspondiente al primero de los datos alfanuméricos, ignorando los demás.

```
PRINT VAL ("7")             retornará 7
PRINT VAL ("MSX")           retornará 0
PRINT VAL ("HOY ES 11")     retornará 0
```

variable Una variable es un nombre dado a un espacio de memoria que puede contener un valor numérico o alfanumérico.
Haciendo referencia al nombre de la variable, obtenemos el valor y podemos cambiarlo.
El nombre de la variable en MSX debe estar formado por letras y números, pero deben empezar por una letra. Solamente son significativos los dos primeros caracteres, aunque puede haber más.
Las variables pueden ser, al igual que las constantes numéricas o alfanuméricas. Las numéricas pueden, asímismo, tener varios tipos de precisión.
La declaración de tipo se puede realizar con instrucciones como DEFINT, DEFSNG, DEFDBL o DEFSTR o mediante caracteres específicos que siguen al nombre de la variable.

A$	variable alfanumérica
B%	variable de precisión entera
C!	variable de precisión simple
D#	variable de precisión doble

A mayor precisión, se requiere mayor espacio de almacenamiento, y en general, mayor tiempo para operar.

variable de conjunto Cuando se tienen que almacenar varios datos numéricos o alfanuméricos homogeneos, en vez de utilizar una variable para cada uno, se utiliza una variable de conjunto que con un único nombre y mediante subíndices, pue de almacenarlos todos. (Véase conjunto).

variable del sistema Son variables utilizadas por el sistema operativo del ordenador para realizar cálculos intermedios, para funciones de control o comunicaciones.
Estas variables no son reconocidas por BASIC y solamente pueden alterarse mediante la instrucción POKE o por rutinas en código máquina.
Las distintas rutinas del sistema operativo hacen uso de ellas, así por ejemplo, las variables del sistema FORCLR, BAKCLR y BDRCLR almacenan los colores de primer plano, fondo y borde.
Si alteramos estos valores con POKE y después ejecutamos la rutina del sistema operativo denominada CNGCLR (cambio de color), los nuevos colores serán mostrados en pantalla.
Para ello utilizaremos un programa como el siguiente:

```
10 SCREEN 1
20 POKE &HF3E9,11
30 POKE &HF3EA,13
40 POKE &HF3EB,15
50 DEFUSR=&H62
60 A=USR(X)
```

Como puede apreciarse, las direcciones &HF3E9, &HF3EA y &HF3EB corresponden a FORCLR, BARCLR y BDRCLR respectivamente, a los que asignamos los colores amarillo claro, magenta y blanco.

VARPTR Es una función BASIC que utiliza como argumento un nombre de variable y proporciona la dirección de memoria a partir de la que se encuentra almacenada la variable.
Si el valor obtenido es negativo (debido a la notación de complemento de dos) hay que sumarle 65536 para obtener el valor real.
VARPTR se puede utilizar también con variables de conjunto, en cuyo caso conviene utilizar como argumento el elemento más bajo del conjunto (subíndices = cero).
También permite conocer la dirección en memoria de un bloque de control de archivo, utilizando como argumento el signo # seguido por el número con que fue abierto el archivo.
En el siguiente ejemplo, se asigna la variable entera A% con el valor 10 y después se imprime el contenido de las posiciones de memoria donde se encuentra almacenada, obteniéndose el valor 10 en la primera y 0 en la segunda.

```
10 A%=10
20 PRINT PEEK(VARPTR(A%))
30 PRINT PEEK(VARPTR(A%)+1)
```

VDP (1) Es una variable especial de BASIC que permite acceder (lectura y escritura) a los registros del procesador de video.
Puede emplearse como una función para leer los registros del VDP o como instrucción para asignar un valor a los registros de control.
En el primer caso se utiliza con el argumento del número de registro que queremos leer.

```
PRINT VDP(8)
```

Imprime el valor del registro 8.

En el segundo caso se asigna un valor a VDP seguida por el número de registro encerrado entre paréntesis.
El siguiente programa desactiva la pantalla temporalmente y después vuelve a activarla haciendo uso del bit 6 del registro 1 del VDP.

```
10 A=VDP(1)
20 VDP(1)=A AND &B10111111
30 FOR N=1 TO 10000: NEXT N
40 VDP(1)=A
```

(2) Son las abreviaturas en inglés del procesador de video, que en la especificación MSX, es el circuito 9129 de Texas Instruments.
El VDP se encarga de la gestión de la pantalla en sus diferentes modos, de los sprites y de la generación de las interrupciones del sistema.
Para ello dispone de una memoria independiente (VRAM) de 16K.
Consta de 8 registros de control (sólo para escritura) y un registro de estado (sólo para lectura) con las funciones siguientes:

Registro 0:
- Bit 0 se usa para activar un VDP externo.
- Bit 1 es el tercer bit de modo M3, que se utiliza conjuntamente con los bits 3 y 4 del registro 1.
- Los restantes bits de este registro no se utilizan.

Registro 1:
- Bit 0 se utiliza para la ampliación de sprites.
  0: tamaño normal
  1: tamaño ampliado
- Bit 1 se utiliza para determinar el tamaño de los sprites.
  0: sprites de 8x8
  1: sprites de 16x16
- Bit 2 debe ser cero.
- Bit 3 es el segundo bit de modo M2.
- Bit 4 es el primer bit de modo M1 juntamente con el bit 1 del registro 0.
  Deben tomar los siguientes valores:

M3	M2	M1	
0	0	0	Texto 32 columnas
1	0	0	Gráficos alta resoluc.
0	1	0	Gráficos baja resoluc.
0	0	1	Texto 40 columnas

- Bit 5 controla las interrupciones.
  0: desactivación
  1: activación
- Bit 6 controla la visualización de la pantalla.
  0: pantalla oscura
  1: pantalla activa
- Bit 7 controla el tamaño de la VRAM
  0: VRAM 4K
  1: VRAM 16K
  Debe ser siempre 1.

Registro 2: Los cuatro bits menos significativos representan los cuatro bits más significativos de la dirección de la tabla de nombres de patrones.

Registro 3: Los ocho bits de este registro constituyen los ocho bits más significativos de la dirección de la tabla de color, que consta de 14 bits.

Registro 4: Los tres bits menos significativos son los tres bits más significati-

vos de la dirección de la tabla de patrones generadores.

Registro 5: Los siete bits menos significativos constituyen los siete bits más significativos de la tabla de atributos de sprite.

Registro 6: Los tres bits menos significativos son los tres bits más significativos de la dirección de la tabla de patrones de sprite.

Registro 7: Este registro contiene en sus cuatro bits más significativos el código de color de texto y en los cuatro bits menos significativos el código de color de fondo.

Registro de estado:

Es un registro de sólo lectura.

- Bit 0 - Bit 4: contienen el número del sprite que ha roto la regla del quinto sprite, por la cual no puede haber más de cuatro sprites en cada línea horizontal.
- Bit 5: este bit toma el valor 1 si dos sprites están solapados, es la bandera de colisión de sprites.
- Bit 6: Adquiere el valor 1 si existen cinco o más sprites en una línea horizontal, es la bandera del quinto sprite.
- Bit 7: este bit señala las interrupciones, tomando el valor 1 al finalizar el barrido de pantalla.

velocidad de transferencia Es la velocidad con que el ordenador es capaz de transmitir información a otros dispositivos.

Se mide en baudios, que equivale a 1 bit/seg. aproximadamente.
La velocidad de transferencia en MSX al cassette es de 1200 ó 2400 baudios, y se selecciona mediante la instrucción SCREEN.
La velocidad de transferencia a la unidad de disco es muy superior, alcanzando los 250 kbaudios, es decir, 250 Kbits/seg.

verdadero Es una constante lógica.
Una expresión lógica como A mayor que cero, puede ser verdadera o falsa.
El sistema MSX representa estos estados lógicos con valores numéricos:

verdadero equivale a -1
falso equivale a 0

En el siguiente programa se asigna un valor a la variable A y se imprimen los valores numéricos asignados a dos expresiones lógicas, la primera verdadera y la segunda falsa.

```
10 A=10
20 PRINT A<>0
30 PRINT A=0
```

Loa valores obtenidos son, respectivamente:

-1 (verdadero)
0 (falso)

verificación de programas
(Véase comprobación).

video compuesto Las señales para el control de pantalla pueden ser de tres tipos:

- Radiofrecuencia: es la señal utilizada por los aparatos de televisión. En el sistema MSX esta señal es del tipo UHF y permite utilizar cualquier TV introduciendo la señal del ordenador por la toma de antena.

- Video compuesto: es una señal que contiene tres señales de color y una de sincronismo mezcladas. Algunos aparatos de TV disponen de una toma separada para este tipo de señal, que es la normalmente utilizada en monitores.
  Precisa de una conexión independiente para el sonido.
  Da una calidad de imagen superior a la RF al evitar circuitos intermedios de una TV.
- RGB: las señales de color y sincronismo viajan independientemente. Precisa de una toma complementaria de sonido.
  Da la máxima calidad posible.
  No todos los monitores la admiten y tampoco todos los ordenadores MSX son capaces de producirla.

visualización de teclas de función
El contenido de las teclas de función se visualiza en la última línea de las pantallas de texto.
Normalmente se muestran los valores de las teclas F1 a F5.
Si pulsamos SHIFT se muestran los de las teclas F6 a F10.
Tras la reposición del sistema, los valores asignados son:

F1	color
F2	auto
F3	goto
F4	list
F5	run + RETURN
F6	color 15,4,4 + RETURN
F7	cload"
F8	cont + RETURN
F9	list + RETURN
F10	SHIFT + HOME + run + RETURN

RETURN, SHIFT y HOME corresponden a las acciones desarrolladas al pulsar dichas teclas en la secuencia expresada.
El contenido de las teclas de función puede visualizarse con KEY LIST.
En algunos casos hay que imprimir en la línea inferior de pantalla, para ello se deben desconectar las teclas de función mediante KEY OFF. Para volverlas a conectar utilizamos KEY ON.
Para redefinir estas teclas se emplea:

KEY nº de tecla, "cadena de definición"

Por ejemplo, si queremos que la tecla F1 nos muestre la pantalla en negro, con letras blancas, hacemos:

```
KEY 1,"color 15,1,1"+CHR$(13)
```

En este caso, CHR$(13) hace las veces de la tecla RETURN.
En modo de texto de 40 columnas se visualizan 6 caracteres de cada tecla, y en el modo de 32 columnas 5 caracteres, si bien esta situación se modifica con WIDTH.

volcado de pantalla Consiste en obtener una copia en impresora del contenido de la pantalla.
Por ejemplo, en modo de texto de 40 columnas podríamos utilizar un programa como el siguiente:

```
10 FOR N=0 TO 959 STEP 40
20 A$=""
30 FOR M=0 TO 39
40 A$=A$+CHR$(VPEEK(N+M))
50 NEXT M
60 LPRINT A$
70 NEXT N
```

VPEEK Función que proporciona el contenido de la dirección especificada de la memoria de video.

P=VPEEK(1000)

Se asigna a la variable P el contenido de la dirección 1000 de la VRAM.
La dirección debe estar comprendida entre 0 y 16383.

VRAM Es la memoria de video (Video RAM), donde se almacena la información visualizada en pantalla.
En el sistema MSX, la VRAM es independiente de la RAM de usuario, y tiene una capacidad de 16K y se encuentra dividida en una serie de tablas (tabla de color, de nombres de patrones, de patrones generadores ...)

WAIT Es una instrucción que lee una de las vías de acceso y comprueba que el dato recibido tiene una valor determinado, en ese caso los datos siguientes son recibidos, en caso contrario, prosigue la ejecución del programa.
Utiliza tres parámetros: el número de la vía de acceso y dos expresiones.
El valor leido es operado con la segunda expresión según el operador XOR y el resultado con la primera expresión según el operador AND. Si el resultado es cero, se reciben los datos siguientes, en caso contrario, prosigue la ejecución del programa.

WIDTH Una instrucción BASIC que permite determinar el número de columnas en las pantallas de texto.
Los valores máximos admitidos son:

40 Pantalla de texto de 40 columnas
32 Pantalla de texto de 32 columnas

Los valores asignados por el sistema son de 37 columnas para la pantalla de 40, y de 29 para la palabra de 32.
De esta forma se busca evitar la pérdida de información en los bordes de la pantalla.

XOR (1) Un operador lógico binario de BASIC que utiliza como operandos expresiones lógicas o numéricas. En el primer caso, se emplea la siguiente tabla de verdad:

X	Y	X XOR Y
V	V	F
V	F	V
F	V	V
F	F	F

En el segundo caso se opera en la forma expuesta en operaciones lógicas.

(2) Un mnemónico de instrucción Z-80 que ejecuta la operación lógica XOR, en forma similar a BASIC pero tomando como operandos un dato, un registro o el contenido de una posición de memoria y el acumulador. Almacena el resultado en A.

Z-80 Es la Unidad Central de Proceso del Sistema MSX (concretamente el Z-80A).
Su frecuencia de funcionamiento es de 3.58 MHz.
Está compuesto por una Unidad Aritmético Lógica, relacionada con el registro A (acumulador) y el registro F (banderas).
Estos registros, junto con los de uso general B, C, D, E, H y L están duplicados con los nombres A', F', B', C', D', E', H' y L'.
Pueden combinarse por pares y comunicarse con el bus de direcciones (de 16 conductores) lo que permite direccionar un total de 64K de memoria.
También hay registros de índice IX, IY, contador de programa PC, puntero de pila SP, interrupción I, refresco de memoria R, registro de instrucción IR.
Consta, asímismo, de una unidad de control, unida al bus del mismo nombre.
Por último, encontramos el bus de datos, que conecta los distintos elementos y junto con los de direcciones y control, compone el bus del procesador.
Además de estos elementos encontramos otros auxiliares, sumadores, multiflexores y buffers.

Z-80 lineas de control Además de los conductores del bus de datos y direcciones, existen una serie de líneas de control. Entre ellas se encuentran:

- RD y WR que representan los estados de lectura o escritura.
- BUSREQ y BUSAK que controlan la conexión de los buses a la CPU.
- MREQ indica que se va a producir una lectura/escritura a memoria.

- IORQ indica que se ha producido una interrupción
- RFSM para indicar el refresco de memoria.
- WAIT detiene el ordenador para completar datos.
- RESET realiza la reposición del sistema.
- INT y NMI señalan las interrupciones enmascarables y no enmascarables.

Z-80 lenguaje máquina El lenguaje del procesador Z-80 es muy extenso. Las distintas instrucciones y operaciones se reconocen mediante los mnemónicos.
Pueden catalogarse en distintos grupos: aritméticas, de carga, lógicas, de rotación y desplazamiento, de bits, relacionadas con subrutinas, de bloques, de control y de Entradas/Salidas.
A continuación se recogen las distintas instrucciones y su código en hexadecimal.

ADC	A,(HL)	8E
ADC	A,(IX+d)	DD8E05
ADC	A,(IY+d)	FD8E05
ADC	A,A	8F
ADC	A,B	88
ADC	A,C	89
ADC	A,D	8A
ADC	A,E	8B
ADC	A,H	8C
ADC	A,L	8D
ADC	A,n	CE20
ADC	HL,BC	ED4A
ADC	HL,DE	ED5A
ADC	HL,HL	ED6A
ADC	HL,SP	ED7A
ADD	A,(HL)	86
ADD	A,(IX+d)	DD8605
ADD	A,(IY+d)	FD8605

ADD	A,A	87
ADD	A,B	80
ADD	A,C	81
ADD	A,D	82
ADD	A,E	83
ADD	A,H	84
ADD	A,L	85
ADD	A,n	C620
ADD	HL,BC	09
ADD	HL,DE	19
ADD	HL,HL	29
ADD	HL,SP	39
ADD	IX,BC	DD09
ADD	IX,DE	DD19
ADD	IX,IX	DD29
ADD	IX,SP	DD39
ADD	IY,BC	FD09
ADD	IY,DE	FD19
ADD	IY,IY	FD29
ADD	IY,SP	FD39
AND	(HL)	A6
AND	(IX+d)	DDA605
AND	(IY+d)	FDA605
AND	A	A7
AND	B	A0
AND	C	A1
AND	D	A2
AND	E	A3
AND	H	A4
AND	L	A5
AND	n	E620
BIT	0,(HL)	CB46
BIT	0,(IX+d)	DDCB0546
BIT	0,(IY+d)	FDCB0546
BIT	0,A	CB47
BIT	0,B	CB40

BIT	0,C	CB41
BIT	0,D	CB42
BIT	0,E	CB43
BIT	0,H	CB44
BIT	0,L	CB45
BIT	1,(HL)	CB4E
BIT	1,(IX+d)	DDCB054E
BIT	1,(IY+d)	FDCB054E
BIT	1,A	CB4F
BIT	1,B	CB48
BIT	1,C	CB49
BIT	1,D	CB4A
BIT	1,E	CB4B
BIT	1,H	CB4C
BIT	1,L	CB4D
BIT	2,(HL)	CB56
BIT	2,(IX+d)	DDCB0556
BIT	2,(IY+d)	FDCB0556
BIT	2,A	CB57
BIT	2,B	CB50
BIT	2,C	CB51
BIT	2,D	CB52
BIT	2,E	CB53
BIT	2,H	CB54
BIT	2,L	CB55
BIT	3,(HL)	CB5E
BIT	3,(IX+d)	DDCB055E
BIT	3,(IY+d)	FDCB055E
BIT	3,A	CB5F
BIT	3,B	CB58
BIT	3,C	CB59
BIT	3,D	CB5A
BIT	3,E	CB5B
BIT	3,H	CB5C
BIT	3,L	CB5D
BIT	4,(HL)	CB66

BIT	4,(IX+d)	DDCB0566
BIT	4,(IY+d)	FDCB0566
BIT	4,A	CB67
BIT	4,B	CB60
BIT	4,C	CB61
BIT	4,D	CB62
BIT	4,E	CB63
BIT	4,H	CB64
BIT	4,L	CB65
BIT	5,(HL)	CB6E
BIT	5,(IX+d)	DDCB056E
BIT	5,(IY+d)	FDCB056E
BIT	5,A	CB6F
BIT	5,B	CB68
BIT	5,C	CB69
BIT	5,D	C86A
BIT	5,E	CB6B
BIT	5,H	CB6C
BIT	5,L	CB6D
BIT	6,(HL)	CB76
BIT	6,(IX+d)	DDCB057E
BIT	6,(IY+d)	FDCB057E
BIT	6,A	CB77
BIT	6,B	CB70
BIT	6,C	CB71
BIT	6,D	CB72
BIT	6,E	CB73
BIT	6,H	CB74
BIT	6,L	CB75
BIT	7,(HL)	CB7E
BIT	7,(IX+d)	DDCB057E
BIT	7,(IY+d)	FDCB057E
BIT	7,A	CB7F
BIT	7,B	CB78
BIT	7,C	CB79
BIT	7,D	CB7A

BIT	7,E	CB7B
BIT	7,H	CB7C
BIT	7,L	CB7D
CALL	C,nn	DC8405
CALL	M,nn	FC8405
CALL	NC,nn	D48405
CALL	NZ,nn	C48405
CALL	P,nn	F48405
CALL	PE,nn	EC8405
CALL	PO,nn	E48405
CALL	Z,nn	CC8405
CALL	nn	CD8405
CCF		3F
CP	(HL)	BE
CP	(IX+d)	DDBE05
CP	(IY+d)	FDBE05
CP	A	BF
CP	B	B8
CP	C	B9
CP	D	BA
CP	E	BB
CP	H	BC
CP	L	BD
CP	n	FE20
CPD		EDA9
CPDR		EDB9
CPIR		EDB1
CPI		EDA1
CPL		2F
CPL		2F
DAA		27
DEC	(HL)	35
DEC	(IX+d)	DD3505
DEC	(IY+d)	FD3505
DEC	A	3D
DEC	B	05

DEC	BC	0B
DEC	C	0D
DEC	D	15
DEC	DE	1B
DEC	E	1D
DEC	H	25
DEC	HL	2B
DEC	IX	DD2B
DEC	IY	FD2B
DEC	L	2D
DEC	SP	3B
DI		F3
DJNZ	e	102E
EI		FB
EX	(SP),HL	E3
EX	(SP),IX	DDE3
EX	(SP),IY	FDE3
EX	AF,AF'	08
EX	DE,HL	EB
EXX		D9
HALT		76
IM	0	ED46
IM	1	ED56
IM	2	ED5E
IN	A,(C)	ED78
IN	B,(C)	ED40
IN	C,(C)	ED48
IN	D,(C)	ED50
IN	E,(C)	ED58
IN	H,(C)	ED60
IN	L,(C)	ED68
INC	(HL)	34
INC	(IX+d)	DD3405
INC	(IY+d)	FD3405
INC	A	3C
INC	B	04

INC	BC	03
INC	C	0C
INC	D	14
INC	DE	13
INC	E	1C
INC	H	24
INC	HL	23
INC	IX	DD23
INC	IY	FD23
INC	L	2C
INC	SP	33
IN	A,(n)	DB20
IND		EDAA
INDR		EDBA
INI		EDA2
INIR		EDB2
JP	nn	C38405
JP	(HL)	E9
JP	(IX)	DDE9
JP	(IY)	FDE9
JP	C,nn	DA8405
JP	M,nn	FA8405
JP	NC,nn	D28405
JP	NZ,nn	C28405
JP	P,nn	F28405
JP	PE,nn	EA8405
JP	PO,nn	E28405
JP	Z,nn	CA8405
JR	C,e	382E
JR	NC,e	302E
JR	NZ,e	202E
JR	Z,e	282E
JR	e	182E
LD	(BC),A	02
LD	(DE),A	12
LD	(HL),A	77

LD	(HL),B	70
LD	(HL),C	71
LD	(HL),D	72
LD	(HL),E	73
LD	(HL),H	74
LD	(HL),L	75
LD	(HL),n	3620
LD	(IX+d),A	DD7705
LD	(IX+d),B	DD7005
LD	(IX+d),C	DD7105
LD	(IX+d),D	DD7205
LD	(IX+d),E	DD7305
LD	(IX+d),H	DD7405
LD	(IX+d),L	DD7505
LD	(IX+d),n	DD360520
LD	(IY+d),A	FD7705
LD	(IY+d),B	FD7005
LD	(IY+d),C	FD7105
LD	(IY+d),D	FD7205
LD	(IY+d),E	FD7305
LD	(IY+d),H	FD7405
LD	(IY+d),L	FD7505
LD	(IY+d),n	FD360520
LD	(nn),A	328405
LD	(nn),BC	ED438405
LD	(nn),DE	ED538405
LD	(nn),HL	228405
LD	(nn),IX	DD228405
LD	(nn),IY	FD228405
LD	(nn),SP	ED738405
LD	A,(BC)	0A
LD	A,(DE)	1A
LD	A,(HL)	7E
LD	A,(IX+d)	DD7E05
LD	A,(IY+d)	FD7E05
LD	A,(nn)	3A8405

LD	A,A	7F
LD	A,B	78
LD	A,C	79
LD	A,D	7A
LD	A,E	7B
LD	A,H	7C
LD	A,I	ED57
LD	A,L	7D
LD	A,n	3E20
LD	A,R	ED5F
LD	B,(HL)	46
LD	B,(IX+d)	DD4605
LD	B,(IY+d)	FD4605
LD	B,A	47
LD	B,B	40
LD	B,C	41
LD	B,D	42
LD	B,E	43
LD	B,H	44
LD	B,L	45
LD	B,n	0620
LD	BC,(nn)	ED4B8405
LD	BC,nn	018405
LD	C,(HL)	4E
LD	C,(IX+d)	DD4E05
LD	C,(IY+d)	FD4E05
LD	C,A	4F
LD	C,B	48
LD	C,C	49
LD	C,D	4A
LD	C,E	4B
LD	C,H	4C
LD	C,L	4D
LD	C,n	0E20
LD	D,(HL)	56
LD	D,(IX+d)	DD5605

LD	D,(IY+d)	FD5605
LD	D,A	57
LD	D,B	50
LD	D,C	51
LD	D,D	52
LD	D,E	53
LD	D,H	54
LD	D,L	55
LD	D,n	1620
LD	DE,(nn)	ED5B8405
LD	DE,nn	118405
LD	E,(HL)	5E
LD	E,(IX+d)	DD5E05
LD	E,(IY+d)	FD5E05
LD	E,A	5F
LD	E,B	58
LD	E,C	59
LD	E,D	5A
LD	E,E	5B
LD	E,H	5C
LD	E,L	5D
LD	E,n	1E20
LD	H,(HL)	66
LD	H,(IX+d)	DD6605
LD	H,(IY+d)	FD6605
LD	H,A	67
LD	H,B	60
LD	H,C	61
LD	H,D	62
LD	H,E	63
LD	H,H	64
LD	H,L	65
LD	H,n	2620
LD	HL,(nn)	2A8405
LD	HL,nn	218405
LD	I,A	ED47

LD	IX,(nn)	DD2A8405
LD	IX,nn	DD218405
LD	IY,(nn)	FD2A8405
LD	IY,nn	FD218405
LD	L,(HL)	6E
LD	L,(IX+d)	DD6E05
LD	L,(IY+d)	FD6E05
LD	L,A	6F
LD	L,B	68
LD	L,C	69
LD	L,D	6A
LD	L,E	6B
LD	L,H	6C
LD	L,L	6D
LD	L,n	2E20
LD	R,A	ED4F
LD	SP,(nn)	ED7B8405
LD	SP,HL	F9
LD	SP,IX	DDF9
LD	SP,IY	FDF9
LD	SP,nn	318405
LDD		EDA8
LDDR		EDB8
LDI		EDA0
LDIR		EDB0
NEG		ED44
NOP		00
OR	(HL)	B6
OR	(IX+d)	DDB605
OR	(IY+d)	FDB605
OR	A	B7
OR	B	B0
OR	C	B1
OR	D	B2
OR	E	B3
OR	H	B4

OR	L	B5
OR	n	F620
OTDR		ED8B
OTIR		EDB3
OUT	(C),A	ED79
OUT	(C),B	ED41
OUT	(C),C	ED49
OUT	(C),D	ED51
OUT	(C),E	ED59
OUT	(C),H	ED61
OUT	(C),L	ED69
OUT	(n),A	D320
OUTD		EDAB
OUTI		EDA3
POP	AF	F1
POP	BC	C1
POP	DE	D1
POP	HL	E1
POP	IX	DDE1
POP	IY	FDE1
PUSH	AF	F5
PUSH	BC	C5
PUSH	DE	D5
PUSH	HL	E5
PUSH	IX	DDE5
PUSH	IY	FDE5
RES	0,(HL)	CB86
RES	0,(IX+d)	DDCB0586
RES	0,(IY+d)	FDCB0586
RES	0,A	CB87
RES	0,B	CB80
RES	0,C	CB81
RES	0,D	CB82
RES	0,E	CB83
RES	0,H	CB84
RES	0,L	CB85

RES	1,(HL)	CB8E
RES	1,(IX+d)	DDCB058E
RES	1,(IY+d)	FDCB058E
RES	1,A	CB8F
RES	1,B	CB88
RES	1,C	CB89
RES	1,D	CB8A
RES	1,E	CB8B
RES	1,H	CB8C
RES	1,L	CB8D
RES	2,(HL)	CB96
RES	2,(IX+d)	DDCB0596
RES	2,(IY+d)	FDCB0596
RES	2,A	CB97
RES	2,B	CB90
RES	2,C	CB91
RES	2,D	CB92
RES	2,E	CB93
RES	2,H	CB94
RES	2,L	CB95
RES	3,(HL)	CB9E
RES	3,(IX+d)	DDCB059E
RES	3,(IY+d)	FDCB059E
RES	3,A	CB9F
RES	3,B	CB98
RES	3,C	CB99
RES	3,D	CB9A
RES	3,E	CB9B
RES	3,H	CB9C
RES	3,L	CB9D
RES	4,(HL)	CBA6
RES	4,(IX+d)	DDCB05A6
RES	4,(IY+d)	FDCB05A6
RES	4,A	CBA7
RES	4,B	CBA0
RES	4,C	CBA1

RES	4,D	CBA2
RES	4,E	CBA3
RES	4,H	CBA4
RES	4,L	CBA5
RES	5,(HL)	CBAE
RES	5,(IX+d)	DDCB05AE
RES	5,(IY+d)	FDCB05AE
RES	5,A	CBAF
RES	5,B	CBA8
RES	5,C	CBA9
RES	5,D	CBAA
RES	5,E	CBAB
RES	5,H	CBAC
RES	5,L	CBAD
RES	6,(HL)	CBB6
RES	6,(IX+d)	DDCB05B6
RES	6,(IY+d)	FDCB05B6
RES	6,A	CBB7
RES	6,B	CBB0
RES	6,C	CBB1
RES	6,D	CBB2
RES	6,E	CBB3
RES	6,H	CBB4
RES	6,L	CBB5
RES	7,(HL)	CBBE
RES	7,(IX+d)	DDCB05BE
RES	7,(IY+d)	FDCB05BE
RES	7,A	CBBF
RES	7,B	CBB8
RES	7,C	CBB9
RES	7,D	CBBA
RES	7,E	CBBB
RES	7,H	CBBC
RES	7,L	CBBD
RET		C9
RET	C	D8

RET	M	F8
RET	NC	D0
RET	NZ	C0
RET	P	F0
RET	PE	E8
RET	PO	E0
RET	Z	C8
RETI		ED4D
RETN		ED45
RL	(HL)	CB16
RL	(IX+d)	DDCB0516
RL	(IY+d)	FDCB0516
RL	A	CB17
RL	B	CB10
RL	C	CB11
RL	D	CB12
RL	E	CB13
RL	H	CB14
RL	L	CB15
RLA		17
RLC	(HL)	CB06
RLC	(IX+d)	DDCB0506
RLC	(IY+d)	FDCB0506
RLC	A	CB07
RLC	B	CB00
RLC	C	CB01
RLC	D	CB02
RLC	E	CB03
RLC	H	CB04
RLC	L	CB05
RLCA		07
RLD		ED6F
RR	(HL)	CB1E
RR	(IX+d)	DDCB051E
RR	(IY+d)	FDCB051E
RR	A	CB1F

RR	B	CB18
RR	C	CB19
RR	D	CB1A
RR	E	CB1B
RR	H	CB1C
RR	L	CB1D
RRA		1F
RRC	(HL)	CB0E
RRC	(IX+d)	DDCB050E
RRC	(IY+d)	FDCB050E
RRC	A	CB0F
RRC	B	CB08
RRC	C	CB09
RRC	D	CB0A
RRC	E	CB0B
RRC	H	CB0C
RRC	L	CB0D
RRCA		0F
RRD		ED67
RST	00H	C7
RST	08H	CF
RST	10H	D7
RST	18H	DF
RST	20H	E7
RST	28H	EF
RST	30H	F7
RST	38H	FF
SBC	A,n	DE20
SBC	A,(HL)	9E
SBC	A,(IX+d)	DD9E05
SBC	A,(IY+d)	FD9E05
SBC	A,A	9F
SBC	A,B	98
SBC	A,C	99
SBC	A,D	9A
SBC	A,E	9B

SBC	A,H	9C
SBC	A,L	9D
SBC	HL,BC	ED42
SBC	HL,DE	ED52
SBC	HL,HL	ED62
SBC	HL,SP	ED72
SCF		37
SET	0,(HL)	CBC6
SET	0,(IX+d)	DDCB05C6
SET	0,(IY+d)	FDCB05C6
SET	0,A	CBC7
SET	0,B	CBC0
SET	0,C	CBC1
SET	0,D	CBC2
SET	0,E	CBC3
SET	0,H	CBC4
SET	0,L	CBC5
SET	1,(HL)	CBCE
SET	1,(IX+d)	DDCB05CE
SET	1,(IY+d)	FDCB05CE
SET	1,A	CBCF
SET	1,B	CBC8
SET	1,C	CBC9
SET	1,D	CBCA
SET	1,E	CBCB
SET	1,H	CBCC
SET	1,L	CBCD
SET	2,(HL)	CBD6
SET	2,(IX+d)	DDCB05D6
SET	2,(IY+d)	FDCB05D6
SET	2,A	CBD7
SET	2,B	CBD0
SET	2,C	CBD1
SET	2,D	CBD2
SET	2,E	CBD3
SET	2,H	CBD4

SET	2,L	CBD5
SET	3,(HL)	CBDE
SET	3,(IX+d)	DDCB05DE
SET	3,(IY+d)	FDCB05DE
SET	3,A	CBDF
SET	3,B	CBD8
SET	3,C	CBD9
SET	3,D	CBDA
SET	3,E	CBDB
SET	3,H	CBDC
SET	3,L	CBDD
SET	4,(HL)	CBE6
SET	4,(IX+d)	DDCB05E6
SET	4,(IY+d)	FDCB05E6
SET	4,A	CBE7
SET	4,B	CBE0
SET	4,C	CBE1
SET	4,D	CBE2
SET	4,E	CBE3
SET	4,H	CBE4
SET	4,L	CBE5
SET	5,(HL)	CBEE
SET	5,(IX+d)	DDCB05EE
SET	5,(IY+d)	FDCB05EE
SET	5,A	CBEF
SET	5,B	CBE8
SET	5,C	CBE9
SET	5,D	CBEA
SET	5,E	CBEB
SET	5,H	CBEC
SET	5,L	CBED
SET	6,(HL)	CBF6
SET	6,(IX+d)	DDCB05F6
SET	6,(IY+d)	FDCB05F6
SET	6,A	CBF7
SET	6,B	CBF0

SET	6,C	CBF1
SET	6,D	CBF2
SET	6,E	CBF3
SET	6,H	CBF4
SET	6,L	CBF5
SET	7,(HL)	CBFE
SET	7,(IX+d)	DDCB05FE
SET	7,(IY+d)	FDCB05FE
SET	7,A	CBFF
SET	7,B	CBF8
SET	7,C	CBF9
SET	7,D	CBFA
SET	7,E	CBFB
SET	7,H	CBFC
SET	7,L	CBFD
SLA	(HL)	CB26
SLA	(IX+d)	DDCB0526
SLA	(IY+d)	FDCB0526
SLA	A	CB27
SLA	B	CB20
SLA	C	CB21
SLA	D	CB22
SLA	E	CB23
SLA	H	CB24
SLA	L	CB25
SRA	(HL)	CB2E
SRA	(IX+d)	DDCB052E
SRA	(IY+d)	FDCB052E
SRA	A	CB2F
SRA	B	CB28
SRA	C	CB29
SRA	D	CB2A
SRA	E	CB2B
SRA	H	CB2C
SRA	L	CB2D
SRL	(HL)	CB3E

SRL	(IX+d)	DDCB053E
SRL	(IY+d)	FDCB053E
SRL	A	CB3F
SRL	B	CB38
SRL	C	CB39
SRL	D	CB3A
SRL	E	CB3B
SRL	H	CB3C
SRL	L	CB3D
SUB	(HL)	96
SUB	(IX+d)	DD9605
SUB	(IY+d)	FD9605
SUB	A	97
SUB	B	90
SUB	C	91
SUB	D	92
SUB	E	93
SUB	H	94
SUB	L	95
SUB	n	D620
XOR	(HL)	AE
XOR	(IX+d)	DDAE05
XOR	(IY+d)	FDAE05
XOR	A	AF
XOR	B	A8
XOR	C	A9
XOR	D	AA
XOR	E	AB
XOR	H	AC
XOR	L	AD
XOR	n	EE20

MICRO-GUIAS
MSX

Explicación detallada de todas las palabras relacionadas con el sistema MSX.

De la A a la Z:

- Conceptos Generales.
- Lenguaje MSX-Basic.
- Código Máquina Z-80.
- Sistema operativo MSX-DOS.



CARRETERA DE CANILLAS, 144 · 28043 MADRID

ISBN 84-86381-10-X

